AN APPROACH TOWARDS DESIGNING

MULTIPLE SECRET STEGANOGRAPHY TECHNIQUE

A Thesis submitted to
The Faculty of Engineering & Technology
Jadavpur University
for the Degree of
Master of Technology in Computer Technology
Dept. of Computer Science & Engineering
Jadavpur University

Thesis Submitted On: May, 2014

By

KRISHNENDU SAHA

Registration No: 116907 of 2011-12
Exam Roll No:M6TCT14-12

Under the esteemed guidance of
Prof. ATAL CHAUDHURI

Dept. of Computer Science & Engineering
Jadavpur University, Kolkata 700032

Faculty of Engineering and Technology

Jadavpur University

Jadavpur

Kolkata-700032

Phone: 91-33-24146666/6194/6643

Website: www.jadavpur.edu.in

Certificate of Approval

This is to certify that the thesis entitled **"AN APPROACH TOWARDS DESIGNING MULTIPLE SECRET STEGANOGRAPHY TECHNIQUE"** is a bona-fide record of work carried out by **Krishnendu Saha (Registration no. 116907 of 2011-12, Exam Roll no. M6TCT14-12)** in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Technology in the department of Computer Science and Engineering, Jadavpur University during the period June 2013 to May 2014. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

(Signature of the Examiner)

(Signature of the Supervisor)

Faculty of Engineering and Technology

Jadavpur University

Jadavpur

Kolkata-700032

Phone: 91-33-24146666/6194/6643

Website: www.jadavpur.edu.in

To whom it may concern

This is to certify that the work embodied in this thesis entitled *"***AN APPROACH TOWARDS DESIGNING MULTIPLE SECRET STEGANOGRAPHY TECHNIQUE***"* has been satisfactorily completed by **Krishnendu Saha (Registration no. 116907 of 2011-12, Exam Roll no. M6TCT14-12)**. It is a bona-fide piece of work carried out at Jadavpur University, Kolkata, for partial fulfillment of requirements for the awarding of the Master of Computer Technology degree of the Department of Computer Science and Engineering, Faculty of Engineering and Technology, Jadavpur University, during the academic year 2011-14.

Prof. Atal Chaudhuri
Dept. of Computer Science & Engg.
Jadavpur University

Prof. Sivaji Bandyopadhay
Head
Dept. of Computer Science & Engg.
Jadavpur University

Prof. Sivaji Bandyopadhay
Dean
Faculty of Engineering & Technology
Jadavpur University

Faculty of Engineering and Technology

Jadavpur University

Jadavpur

Kolkata-700032

Phone: 91-33-24146666/6194/6643

Website: www.jadavpur.edu.in

ACKNOWLEDGEMENT

I would like to convey my most sincere gratitude to my Thesis Supervisor, **Prof. ATAL CHAUDHURI** for his constant encouragement, invaluable guidance and support throughout the duration of this project. His motivation always gave me the required inputs and momentum to continue with my work, without which this project work would not have taken to its current shape. His valuable suggestions and numerous discussions have always inspired new ways of thinking. My heart filled gratitude goes to him for giving me the idea for working on such an interesting project and helping me to work on it independently. I feel deeply honored that I got this opportunity to work with him.

I would like to thank Prof. Sivaji Bandyopadhay, Head of the Department of Computer Science & Engineering Department and also all other faculty members of the Department of computer Science and Engineering for their continuous support.

Last but not the least, I would like to thank to all the Lab Assistants for providing me lab facility whenever required and my batch mates of Masters of Technology in Computer Technology batch of 2011-14 in Jadavpur University for staying by my side when I needed them. Finally I would like to thank my seniors for providing me useful tips and supports.

Dated:…….

……

Krishnendu Saha
Registration No.-116907 of 2011-2012
Class Roll No- 001110504013
Exam. Roll No. – M6TCT14-12
JADAVPUR UNIVERSITY
KOLKATA-700032

Faculty of Engineering and Technology

Jadavpur University

Jadavpur

Kolkata-700032

Phone: 91-33-24146666/6194/6643

Website: www.jadavpur.edu.in

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of my degree Master of Technology in Computer Technology studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name	:	KRISHNENDU SAHA
Roll Number	:	M6TCT14-12
Thesis Tile	:	AN APPROACH TOWARDS DESIGNING MULTIPLE SECRET STEGANOGRAPHY TECHNIQUE
Signature	:	
Date	:	

Table of Contents

1 Abstract

It’s very common practice now a days for computer users to exchange private information over internet. To ensure the security of exchanged information, the most common way is to transform the data into a different form so that the resulting data can be understood only by those who know how to get back the data in the original form. This method of protecting information is known as encryption. A major drawback of this mechanism is that the existence of the transformed data which is not hidden. Presence of this data over digital media, though encrypted, always draws attention to the hackers to do something on that data to access secure private information. One of the possible ways to solve this problem is steganography. This is one of the ancient art of hiding messages so that they are not detectable. Here the messages are kept hidden in such an unsuspecting way that it does not draws any attention to the hackers. Steganography's intent is to hide the existence of the message whereas cryptography transforms a message into different form so that it cannot be understood though the existence of message might be known. Here in this thesis, we tried to apply steganography technique in such a way so that multiple secrets can be shared among a set of participants in different groups in such a way that only qualified sets of participants from a particular group can recover the secrets, whereas non-qualified sets of participants might not be able to derive any meaningful information from it.

The proposed Multiple Secret Steganography scheme is a method to encrypt P number of secrets (say S1, S2,…, Sp) that is to be shared among P different groups (say G1, G2,…,Gp). Here each group may consists different number of members say N1, N2, …., Np and there exists a threshold (say k1, k2, …, kp) for each group such that secret data for a particular group can be reconstructed from the collection of shares only when number of shares are greater than or equal to the threshold value for that group.

In this Group Secret sharing scheme, first the secret data for each group is encrypted and distributed {(S11, S12,…,S1N), (S21, S22,…S2N)...(Sp1,Sp2,….,SpN)} into N (where N is the maximum among N1,N2,…,Np) intermediate shares using Secret Sharing Scheme (SSS). Then shares are taken one by one from each group and encapsulated into envelope images applying Steganography Technique. So first envelope image has all the first shares (S11,S21,…,Sp1) from each group, second envelope image has all the second shares (S12,S22,…,Sp2) and so on for rest of the envelope images. Apart from this, all the groups have specific Group Keys (say gK1,gK2,…,gKp) which is used for secure encapsulation of intermediate shares in the envelope images. This way, all the shares are encapsulated into N different envelope images in a secure manner based on respective group keys. Now these N Steganography images (encrypted data encapsulated within it) are final shares which are shared among members of each groups.

On the other hand, reconstruction or decryption of data for a specific group require two things, a proper Group Key and number of envelope images that is more than or equal to the threshold number (say, k1 for group-1) for that group. If and only if the number of secret shared images are more than or equal to the threshold value of that specific group, then this proposed scheme ensures faithful reconstruction for a specific group based on it group key.

This proposed scheme not only provide mechanism for robust and secured Secret Sharing but also it provides low computational complexity during both sharing and reconstruction phases and this technique can be applied seamlessly irrespective of data formats (e.g. Text, Image, Audio, etc.) of the secret message.

Key words:

SSS – Secret Sharing Scheme,
Threshold – Minimum number of shares requires to reconstruct the secret,
Group key – key that is shared among group members used to reconstruct the group secret,
Steganography – Hiding of secret message within another data such that presence of that secret message is not easily detectable.

2 Introduction

The effective and secure protections of sensitive information are primary concerns in commercial, medical and military system (e.g. communication systems or network storage systems). Needless to say, it is also important for an information fusion process to ensure data is not being tampered. Encryption methods are one of the popular approaches to ensure the integrity and secrecy of the protected information. However, one of the critical vulnerabilities of encryption techniques is single-point-failure. For example, the secret information cannot be recovered if the decryption key is lost or the encrypted content is corrupted during the transmission.

Multiple Secret Steganography scheme deals with such problem, namely sharing a highly sensitive secret among a group of n users so that only when a sufficient number k of them come together, the secret can be reconstructed. It is a protocol to share a number of secrets among a set of participants in different groups in such a way that only qualified sets of participants from a particular group can recover the secrets, whereas non-qualified sets of participants might have partial information about them.

This paper presents a robust and secured Multiple Secret Steganography method which is described in subsequent chapters. Before jump into the details of this proposed scheme, in this chapter I shall briefly touch upon some generic concepts and a few referral works done in the field of Cryptography and Steganography.

2.1 Concept of Cryptography

Cryptography is the science of information security. It is nothing but the art of secret writing. The word is derived from the Greek "kryptos", meaning hidden. Cryptography is closely related to the disciplines of cryptology and cryptanalysis. Cryptography includes techniques such as microdots, merging words with images, and other ways to hide information in storage or transit. However, in today's computer-centric world, cryptography is most often associated with scrambling plaintext (ordinary text, sometimes referred to as clear text) into cipher text (a process called encryption), then back again (known as decryption). Individuals who practice this field are known as cryptographers. Modern cryptography concerns itself with the following four objectives:

- **Confidentiality:** Confidentiality is probably the most common aspect of information security. We need to protect our confidential information. An organization needs to guard against those malicious actions that endanger the confidentiality of its information. In the military, concealment of sensitive information is the major concern. In industry, hiding some information from competitors is crucial to the operation of the organization. In banking, customers account need to be kept secret.

- **Integrity:** Information needs to be changed constantly. In a bank, when a customer deposits or withdraws money, the balance of his/her account needs to be changed. Integrity means that changes need to be done only by authorized entities and through authorized mechanisms. Integrity violation is not necessarily the result of a malicious act; an interruption in the system, such as a power surge, may also create unwanted changes in some information.

- **Non-repudiation**: The creator/sender of the information cannot deny at a later stage his or her intentions in the creation or transmission of the information.

- **Authentication:** The sender and receiver can confirm each other's identity and the origin/destination of the information.

Procedures and protocols that meet some or all of the above criteria are known as cryptosystems. The origin of cryptography is usually dated from about 2000 BC, with the Egyptian practice of hieroglyphics. These consisted of complex pictograms, the full meaning of which was only known to an elite few. The first known use of a modern cipher was by Julius Caesar (100 BC to 44 BC), who did not trust his messengers when communicating with his governors and officers. For this reason, he created a system in

which each character in his messages was replaced by a character three positions ahead of it in the Roman alphabet.

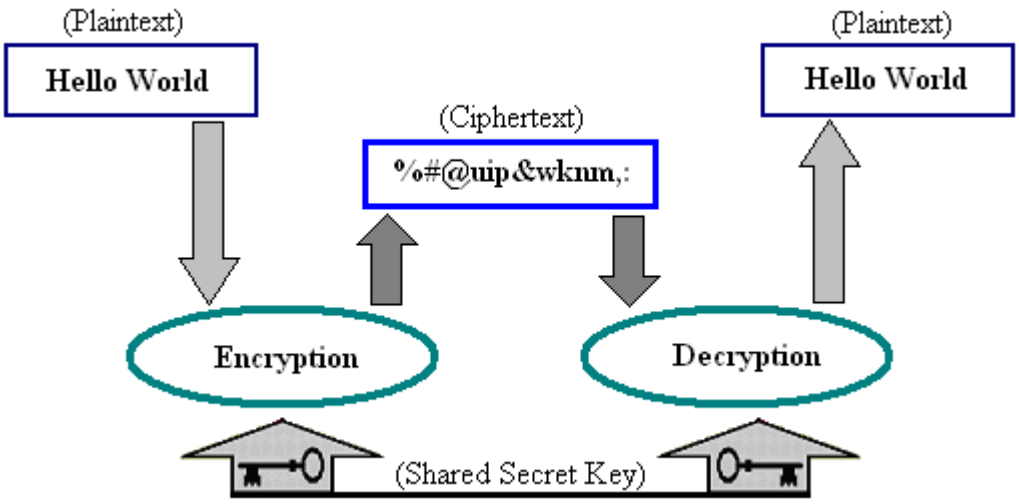
In recent times, cryptography has turned into a battleground of some of the world's best mathematicians and computer scientists. The ability to securely store and transfer sensitive information has proved a critical factor in success in war and business. Modern cryptography intersects the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce.

2.2 General Idea of Cryptography

Cryptography consists of two processes. Firstly, encryption which is the process of converting ordinary information (plaintext) into unintelligible gibberish (i.e., cipher text). Secondly, decryption which is the reverse, in other words, moving from the unintelligible cipher text back to plaintext, as shown in following figure.

A cipher (or cipher) is a pair of algorithms for performing the process of encryption and decryption. The detailed operation of a cipher is controlled both by the algorithm and in each instance by a key. This is a secret parameter (ideally known only to the communicants) for a specific message exchange context.

The key is the most important part in the process of cryptography. If it gets corrupted or lost after the process of encryption then the secret message cannot be decrypted.



The process of Cryptography

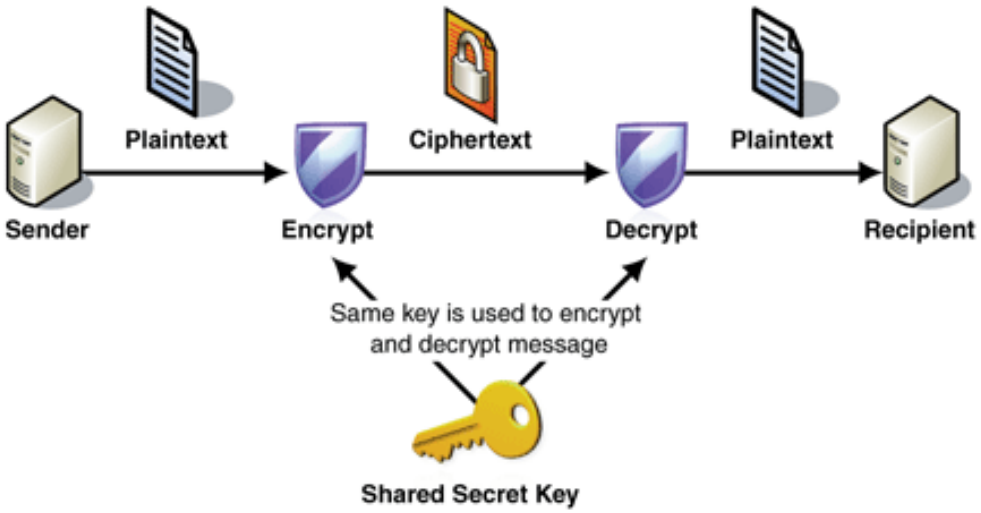
Thus three characters participate in an information exchange scenario; the sender who needs to send secure data, the receiver who is the recipient of the data and the person who somehow disturbs the communication between sender and receiver by intercepting messages to uncover the data or by sending her own disguised message.

2.3 Various types of Cryptography

Cryptography involves scrambling a message or creating a digest of the message. Various types of Cryptographic techniques are discussed briefly in following section.

2.3.1 Symmetric- Key Cryptography

Symmetric-key cryptographic started thousands of years ago when people needed to exchange secrets (for example in a war).We still mainly use symmetric-key cryptography in our network security. In symmetric-key cryptography, the same key is used by the sender as well as the receiver. The sender uses this key and an encryption algorithm to encrypt data; the receiver uses the same key and corresponding decryption algorithm to decrypt the data. The process is shown in figure below.

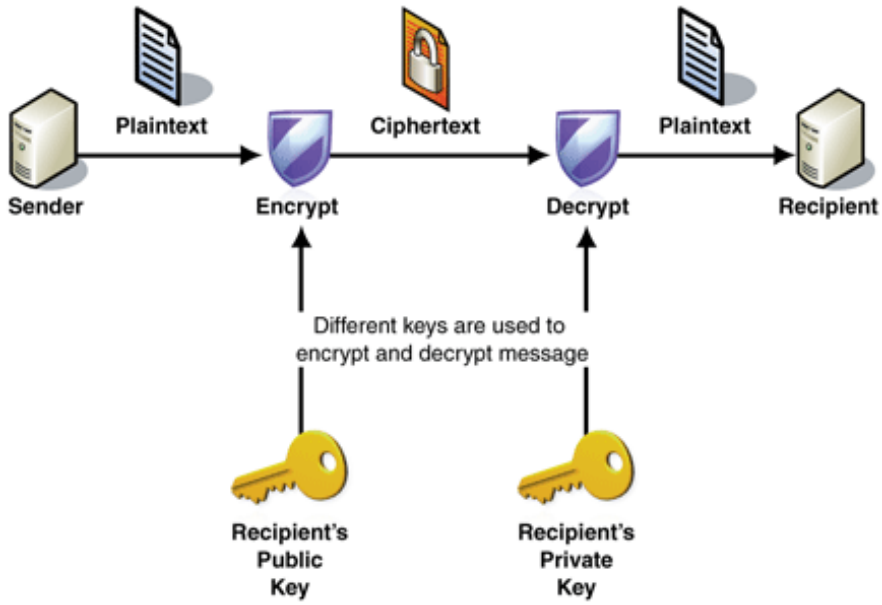


Symmetric-Key Cryptography

2.3.2 Asymmetric-key Cryptography

In Asymmetric or public-key cryptography, there are two keys: a private key and a public key. The private key is kept by the receiver. The public key is announced to the public. The

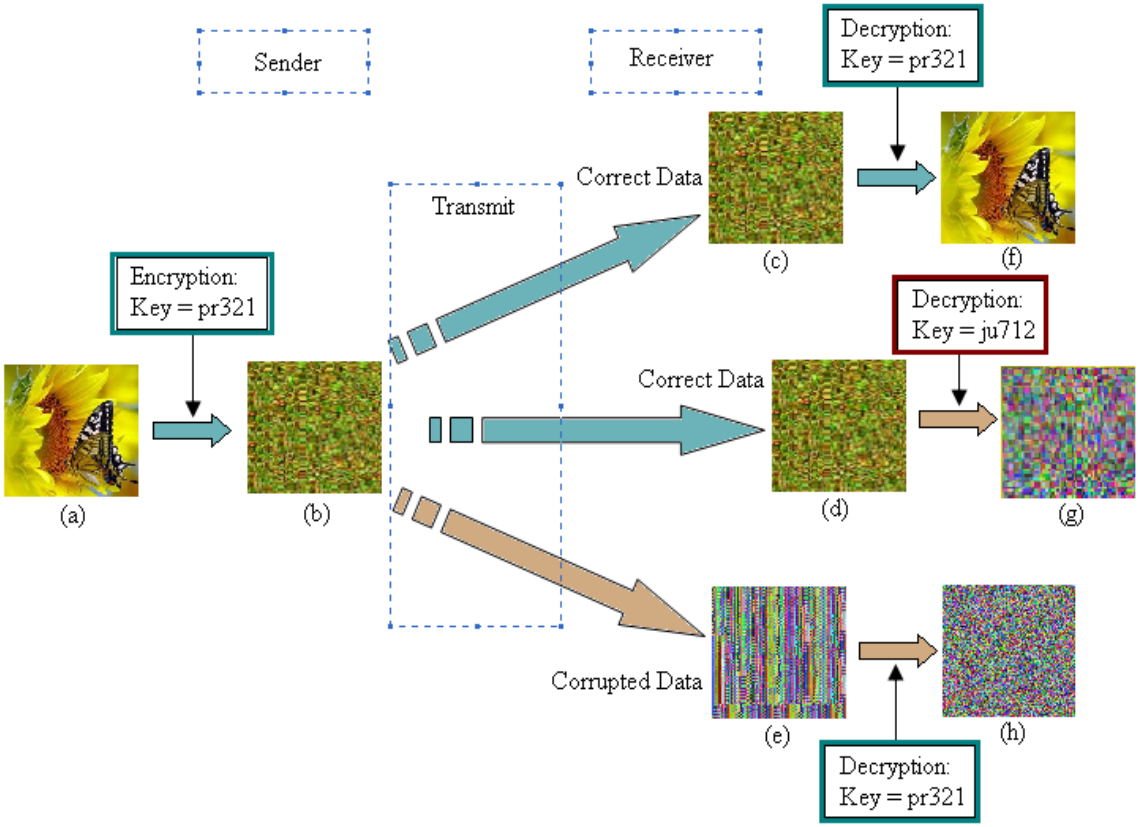
public key is used by the sender to encrypt the message while the private key kept with the receiver to decrypt the message. The process is shown in figure below.



Asymmetric – Key Cryptography

2.4 Drawbacks of Traditional Cryptography

Protection of sensitive data is an important issue, during transmission over internet. Many cryptographic techniques are there to protect secret data. However, a common weakness of this technique is that an entire secret data is kept in single medium. The secret data cannot be revealed if the medium or key is lost or corrupted. This is termed as single point failure. Following figure shows sender encrypts a secret image using the key 'pr321' and then transmits to the receiver. At the receiver end, receiver may receive correct data or data may be corrupted during transmission. In this situation, using correct key reconstruction will be lossless for correct received data otherwise reconstruction will not be possible.



(a) Secret image, (b) Encrypted image using key, (c) & (d) Received correct data, (e) Received corrupted data, (f) Lossless decrypted image using correct key, (g) Complete noisy image using corrupted key and (h) Complete noisy image using correct key

Single point failure

In symmetric-key cryptography, the same key is used by the sender as well as the receiver. Here if the key gets lost or corrupted, the secret data cannot be recovered. Again in asymmetric cryptography, a public key and a private key of the receiver is used by the sender and the receiver respectively. So, copies of the public key of a particular receiver need to be known by many senders that increased the risk of compromise.

The secure storage of the private keys in cryptography is an important issue. The possession of an extremely sensitive key by an individual may not be advisable as the key can easily be lost or as the individual may not be fully trusted. Again given copies of key to more than one individual increases the risk of compromise. The obvious solution is to give different shares of the key to several individuals, forcing them to co-operate to find the secret key. This not only reduces the risk of losing the key but also makes compromising the key more difficult. For that reason secret sharing become more popular where the key is shared among a set of participants. So the sole responsibility of the key no longer exists. Each

participant receives a share of the key. Only qualified set of those participants can recover the key.

2.5 General Idea of Threshold Cryptography

Imagine that a group of scientists came up with a formula which must be kept as a secret. Therefore the formula is encrypted and the key is stored in some place which is hopefully safe. The formula is so important for the scientists that they cannot stand losing it. For this reason they make several copies of the encryption key and store it in different locations. But this would increase the risk of key being stolen. What they actually need is somehow split the key into pieces, store the pieces in different locations and be able to reconstruct the key even if some of the pieces are missing. If a piece is lost or stolen, they can recover the key from the remaining pieces yet the stolen piece would not be sufficient to recover the key.

As a second example, that there is a deadly weapon whose production is an automated process controlled by a computer. The product plan has been loaded to the computer and the computer controls the production by executing the instructions given in the plan. The plan is kept encrypted. Before each production the plan is decrypted and after the production, decrypted plan is deleted. This guarantees that the production can only be initiated by authorized people. This weapon is so dangerous that no one is allowed to start its production alone. Therefore the decryption key must be distributed to the officials such that the production can be started only when an authorized group decides to do so. This can be made possible if there is software loaded to the computer which can split the original encryption key into pieces and reconstruct the key using some of these pieces. In this scenario, we assume that the computer does not store the original encryption key permanently and it works perfectly securely and reliably. It is also important for an information process to ensure data is not being tampered. Encryption methods are one of the popular approaches to ensure the integrity and secrecy of protected information. However, one of the critical vulnerabilities of encryption techniques is single-point-failure. That is the secret information cannot be recovered if the decryption key is lost or the encrypted content is corrupted during the transmission. To overcome this drawback, secret sharing becomes more popular.

In the Secret sharing scheme there is one dealer and n players. The dealer gives a secret to the players, but only when specific conditions are fulfilled. The dealer accomplishes this

by giving each player a share in such a way that any group of k or more players reconstruct the secret but no group of less than k players can. Such a system is called a (k, n)-threshold based secret sharing scheme.

Threshold Secret sharing scheme thus says that: A secret is some data D. Our goal is to divide D into n shares $D_1, D_2 ..., D_n$ in such a way that:

1. Knowledge of any k or more Di shares makes D easily computable, where $1 \leq i \leq n$ and $2 < k \leq n$.
2. Knowledge of any k-1 or fewer D_i shares leaves D completely undetermined (in the sense that all its possible values are equally likely).

If k = n, then all the shares are required in the (n, n)-threshold scheme to recover the secret. However, the loss of any of the shares produced using the (n, n)-threshold scheme results in inaccessible secret messages.

Therefore, apart from the simplest (2, 2)-schemes commonly used as a private key cryptosystem solution, general (k, n)-threshold schemes with k < n are often the object of interest due to offer their ability to recover the secret message even if several shares are lost. In this case, any of [n! /(k !(n – k)!)] possible combinations of k shares can be used to recover the secret message.

2.6 Referral Works

Well known secret sharing schemes (SSS) in the literature includes Shamir's SSS[1] based on polynomial interpolation, Blakley's SSS[2] based on hyper plane geometry, Asmuth-Bloom's SSS[3] based on Chinese Remainder theorem.

Karnin et al. [4] suggested the concept of perfect secret sharing (PSS) where zero information of the secret is revealed for an unqualified group of (k-1) or fewer members. Specifically, Karnin et al.[4] used a term referred as information entropy (a measurement of the uncertainty of the secret), denoted as H(s) where s is a secret shared among n participants. The claim of PSS (Perfect Secret Sharing) schemes must satisfy the following:

1. A qualified coalition of n or more participants, C can reconstruct the secret (s), s: $H(s|C) = 0 \quad \forall |C| \geq k$,
2. An unqualified coalition of (n-1) or few participants, C has no information about the secret (s) s: $H(s|C) = H(s) \quad \forall |C| < k$

For these requirements in PSS schemes, a secret has zero uncertainty if the secret can be discovered by n or more participants. On the contrary, the secret, in PSS schemes, remain the same uncertainty for (k-1) or fewer members. Therefore, there is no information exposed to the (k-1) or fewer members.

A shortcoming of above secret sharing schemes is the need to reveal the secret shares during the reconstruction phase. The system would be more secure if the subject function can be computed without revealing the secret shares or reconstructing the secret back. This is known as function sharing problem where the function's computation is distributed according to underlying SSS such that distributed parts of computation are carried out by individual user and then the partial results can be combined to yield the final result without disclosing the individual secrets. Various function sharing protocols are been proposed [5], [6], [7], [8], [9], [10], [11] mostly based on Shamir's secret sharing as the underlying scheme. Some work [12] is also available on Blakley's secret sharing scheme, and Asmuth-Bloom scheme [14] as well. And a better image secret sharing approach was presented by Thien & Lin [13]. With some cryptographic computation, they cleverly used Shamir's SSS to share a secret image.

2.7 Review of secret sharing schemes

Shamir, Blakley, Asmuth-Bloom and Thien and Lin's Secret Sharing Schemes are briefly described in this section with their interesting features.

2.7.1 Shamir's secret sharing scheme

Adi Shamir's [1] secret sharing scheme is based on (k, n)-threshold based secret sharing technique ($k \leq n$). The technique allows any k out of n shares to construct a given secret, a (k-1) degree polynomial is necessary. This polynomial function of order (k-1) is constructed as,

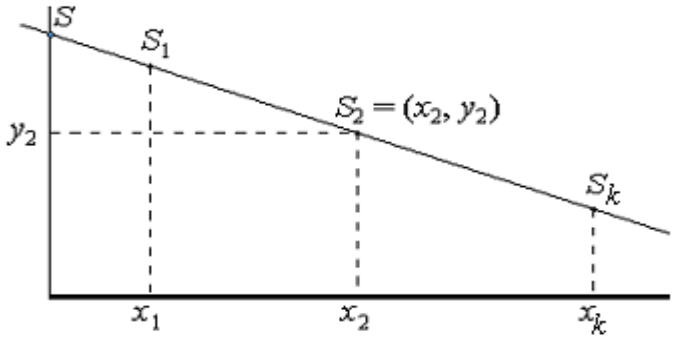
$$F(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + - - - + a_{k-1} x^{k-1} (\text{mod } p)$$

Now we can easily generate n number of shares by using above equation. Where a_0 is the secret, p is a prime number and all other coefficients are random elements from the secret. Each of the n shares is a pair (x_i, y_i) of numbers satisfying $f(x_i) = y_i$ and $x_i > 0$, $1 \leq i \leq n$ and $0 < x_1 < x_2 \ldots < x_k \leq p-1$. Given any k shares, the polynomial is uniquely determined and hence

the secret a_0 can be computed via Lagrange interpolation. However, given k-1 or fewer shares, the secret can be any element in the field.

The polynomial function f(x) is destroyed after each shareholder possesses a pair of values (x_i, y_i) so that no single shareholder knows the secret value a_0. In fact, no groups of k-1 or fewer shares can discover the secret a_0. On the other hand, when k or more secret shares are available, then we may set at least k linear equations $y_i = f(x_i)$ for the unknown a_i's. The unique solution to these equations shows that the secret value a_0 can be easily obtained by using Lagrange interpolation.

Shamir's SSS is regarded as a PSS scheme because knowing even (k-1) linear equations doesn't expose any information about the secret.



Shamir's secret sharing scheme where k=2

Some of the useful properties of Shamir's (k, n)-threshold scheme are:

1. **Secure**: Information theoretic security.
2. **Minimal**: The size of each piece does not exceed the size of the original data.
3. **Extensible**: When n is kept fixed, D_i pieces can be dynamically added or deleted without affecting the other pieces.
4. **Dynamic**: Security can be easily enhanced without changing the secret, but by changing the polynomial occasionally (keeping the same free term) and constructing new shares to the participants.
5. **Flexible**: In organizations where hierarchy is important, we can supply each participant different number of pieces according to his importance inside the organization. For instance, the president can unlock the safe alone, whereas 3 secretaries are required together to unlock it.

2.7.1.1 Example to describe Shamir's secret sharing scheme

The following example describes the basic idea of Shamir's SSS. However, the calculations in the example are done using integer arithmetic rather than using finite field arithmetic. Therefore the example below does not provide perfect secrecy, and is not a true example of Shamir's scheme.

Sharing Phase:

Generating keys using Shamir's scheme is fairly trivial. The original secret is encoded as an integer, for an example we will use 456 as the secret.

The dealer then chooses k-1 random integers, for instance, if k = 3, we pick 173 and 251 at random.

This polynomial is simply the secret $F(x) = a_0 + a_1x_1 + a_2x_2 + - - - + a_{k-1}x_{k-1}$, where a_i (i=1, 2, - - -, k-1) is a random integer and a_0 is secret.

So our polynomial to create the keys is then $f(x) = 456 + 173x + 251x_2$.

The keys themselves are very simple; they are just points on the polynomial. Keys are in the form (x, f(x)), so to create keys we place different values of x into the equation of the polynomial. The first key would be (1, 880), the second (2, 1806), and so on.

If n = 5, our 5 keys will be (1, 880); (2, 1806); (3, 3234); (4, 5164); (5, 7596).

Reconstruction Phase:

Reconstruction phase of Shamir's scheme is not so trivial. This phase involves computing the Lagrange basis polynomials. This manual will not cover Lagrange polynomials in any great depth, but the above example is solved by the following equation:

Assume we receive 3 of the required keys to obtain the secret. These keys are (1, 880), (3, 3234) and (5, 7596).

For this example:

$(x_0, y_0) = (1, 880)$, $(x_1, y_1) = (3, 3234)$, and $(x_2, y_2) = (5, 7596)$

The Lagrange basis polynomial for k = 3 would be:

$$L_0 = \frac{x-x_1}{x_0-x_1} \times \frac{x-x_2}{x_0-x_2} = \frac{x-3}{1-3} \times \frac{x-5}{1-5} = \frac{x^2-8x+15}{8}$$

$$L_1 = \frac{x-x_0}{x_1-x_0} \times \frac{x-x_2}{x_1-x_2} = \frac{x-1}{3-1} \times \frac{x-5}{3-5} = \frac{-x^2+6x-5}{4}$$

$$L_2 = \frac{x-x_0}{x_2-x_0} \times \frac{x-x_1}{x_2-x_1} = \frac{x-1}{5-1} \times \frac{x-3}{5-3} = \frac{x^2-4x+3}{8}$$

We know,

$$f(x) = \sum_{i=0}^{n-1} y_i . L_i$$

Therefore,

$$f(x) = 880(\frac{x^2-8x+15}{8}) + 3234(\frac{-x^2+6x-5}{4}) + 7596(\frac{x^2-4x+3}{8})$$

$f(x) = 251x_2 + 173x + 456$

$f(x) = 456 + 173x + 251x_2$

We now have the original polynomial used to create the keys. The secret is simply the free coefficient, in this case 456.

2.7.2 Blakley's Secret Sharing Scheme

Two nonparallel lines in the same plane intersect at exactly one point. Three "nonparallel" planes in a space intersect at exactly one point. More generally, any n-dimensional hyper planes intersect at a specific point. The secret may be encoded as any single coordinate of the point of intersection. If the secret is encoded using all the coordinates, even if they are random, then an insider (someone in possession of one or more of the n-dimensional hyper planes) gains information about the secret since he knows it must lie on his plane. If an insider can gain any more knowledge about the secret than an outsider can, then the system no longer has information theoretic security. If only one of the n coordinates is used, then the insider knows no more than an outsider (i.e., that the secret must lie on the x-axis for an 2-dimensional system). Each player is given enough information to define a hyper plane; the secret is recovered by calculation the planes point of intersection and then taking a specified coordinate of that intersection.

Blakley's scheme in three dimensions: each share is a plane, and the secret is the point at which three shares intersect. Two shares are insufficient to determine the secret, although they do provide enough information to narrow it down to the line where both planes intersect.

Blakley's[2] scheme is less space-efficient than Shamir's, while Shamir's shares are individually as large as the original secret. This scheme uses hyper plane geometry to solve the secret sharing problem. The secret is a point in a k-dimensional space and n shares are affine hyper planes that pass through this point. An affine hyper plane in a k-dimensional space with coordinates in a field $\mathbb{F}$ can be described by a linear equation of the following form:

$$a_1x_1 + a_2x_2 + - - - + a_kx_k = b$$

The intersection point is obtained by finding the intersection of any k of these hyper planes. The secret can be any of the coordinates of the intersection point or any function of the coordinates. We take the secret to be the first coordinate of the point of intersection.

Sharing Phase:

Let $\mathbb{F}$ be the field we are working on. The dealer generates a secret point x in $\mathbb{F}$, where the first coordinate $x_{[1]}$ is set to the secret value and sets the values of the other coordinates randomly from the field $\mathbb{F}$. The i^{th} user will get a hyper plane equation over $\mathbb{F}$,

$$a_{i1}x_1 + a_{i2}x_2 + - - - + a_{ik}x_k = y_i.$$

For a (k, n)-threshold scheme there will be m such hyper plane equations, and hence we will have an k × n linear system,

$$Ax = y$$

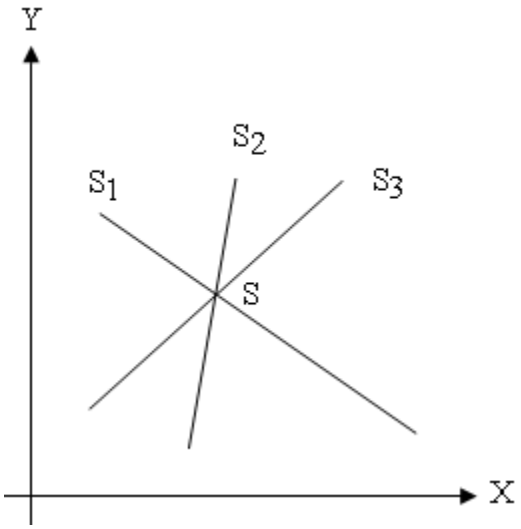
Then sends the secret value of y_i along with $a_{i1, - - -,} a_{ik}$ to user i. The coefficients a_{ij} are not sensitive and can be made public if needed.

Reconstruction Phase:

Share combining step is simply finding the solution of a linear system of equations. Suppose that a coalition S = {i_1, - - -, i_k} of users come together. They form a matrix A_S using their hyper plane equations and solve $A_S x = y_S$

Where, y_S is the vector of the secret shares of the users. The secret is found as the first coordinate of the solution.

For Example, a 2-Dimensional space is considered where three lines intersect at one point as shown in following figure. The each of the three lines represents a secret share and the point of intersection is the secret having unknown x and y co-ordinates.



Blakley's secret sharing scheme in 2-Dimensional Space

So according to the figure drawn above there are three equations for three generated shares. Those equations are as follows:

$A_{11}X + A_{12}Y = V_1$.

$A_{21}X + A_{22}Y = V_2$.

$A_{31}X + A_{32}Y = V_3$.

These are respectively the equations of share S_1, S_2 and S_3.

Since it's a 2–dimensional space, so there are two unknown variables, X and Y, as the unknown co-ordinates for the secret point S, whose value can be known by solving any two of the above three equations. As we get the value of the variables, the secret point is revealed. So for a 2-D space the threshold is two. Thus the threshold value for a n-dimensional space is n. It was an example of a (2, 3)-threshold secret sharing scheme.

2.7.2.1 Example to describe Blakley's secret sharing scheme

Let us consider the Secret value is S = (X, Y) = (120, 150) and the random coefficient values are as follows:

$(A_{11}, A_{12}) = (10, 4)$

$(A_{21}, A_{22}) = (6, 20)$

$(A_{31}, A_{32}) = (15, 10)$

Those coefficient values are not sensitive and made as public.

Sharing Phase:

Here we use (2, 3)-threshold scheme. Therefore 3 shares value are shown below

$V_1 = 10 \times 120 + 4 \times 150 = 1800$

$V_2 = 6 \times 120 + 20 \times 150 = 3720$

$V_3 = 15 \times 120 + 10 \times 150 = 3300$

Therefore the secret value (120, 150) is shared among 3-participants as the value of 1800, 3720 and 3300 respectively.

Reconstruction Phase:

In reconstruction phase, if we collect any 2 shared values we can easily reconstruct the secret value using any linear equation solving techniques (e.g. Cramer's Rule).

Suppose we collect the shared value 1800 and 3300. And we will easily reconstruct the equation by random coefficient value that is made as public.

Therefore the equations will be as follows:

$10 \times X + 4 \times Y = 1800$

$15 \times X + 10 \times Y = 3300$

Using above two equations, we will easily reconstruct the secret value S = (120, 150).

2.7.3 Asmuth-Bloom's Secret Sharing Scheme

A fundamentally different Secret sharing scheme is Asmuth-Bloom's[3] secret sharing scheme which shares a secret among the parties using modular arithmetic and reconstructs it by Chinese Remainder Theorem (CRT). In Asmuth–Bloom Secret Sharing Scheme, the sharing and Reconstruction of the secret can be done as follows:

Sharing Phase:

To share the secret d among a group of n users, the dealer does the following:

a. A set of relatively prime integers $m_0 < m_1 < m_2 < \ldots.. < m_n$ where $m_0 > d$ is a prime, are chosen such that $_{i=1}\Pi^{k}\ m_i > m_0\ \ _{i=1}\Pi^{k-1} m_{n-i+1}$

b. Let M denote $_{i=1}\Pi^{k}\ m_i$, the dealer computes,

$$y = d + am_0$$

where a is a positive integer generated randomly subjected to the condition that $0 \leq y < M$

c. The share of the i^{th} user $1 \leq i \leq n$

$$y_i = y \bmod m_i.$$

Reconstruction Phase:

Assume S is a coalition of k users to reconstruct the secret, let M_s denote $\prod_{i \epsilon S} m_i$

a. Given the system

$$y = y_i \pmod{m_i}.$$

For i belongs to S, solve y in Z_{MS} using the Chinese Remainder Theorem.

b. Compute the secret as:

$$d = y \bmod m_0.$$

According to Chinese Remainder Theorem, y can be determined uniquely in Z_{MS}. Since $y < M \leq M_S$, the solution is also unique in Z_M.

2.7.3.1 Example to describe Asmuth-Bloom's secret sharing scheme

The following example describes the Asmuth-Bloom's Scheme. For practical purposes we choose small values for all parameters.

We choose k=3 and n=4.

Our pair wise co-prime integers being $m_0 = 3$, $m_1 = 11$, $m_2 = 13$, $m_3 = 17$ and $m_4 = 19$.

They satisfy the Asmuth-Bloom required sequence because $3.17.19 < 11.13.17$.

Say our secret S is 2. Pick $\alpha = 51$, satisfying the required condition for the Asmuth-Bloom scheme.

Then $2 + 51 \times 3 = 155$ and we compute the shares for each of the integers 11, 13, 17 and 19. They are respectively 1, 12, 2 and 3. We consider one possible set of 3 shares: among the 4 possible sets of 3 shares we take the set {1, 12, 2} and show that it recovers the secret S=2. Consider the following system of congruencies:

$$x \equiv 1 \bmod 11$$

$$x \equiv 12 \bmod 13$$

$$x \equiv 2 \bmod 17$$

To solve the system, let $M = 11 \times 13 \times 17$.

From a constructive algorithm for solving such a system, we know that a solution to the system is $x_0 = 1 \times e_1 + 12 \times e_2 + 2 \times e_3$, where each e_i is found as follows:

since gcd (m_i, M / m_i) = 1, there exist positive integers r_i and s_i, that can be found using the Extended Euclidean algorithm, that s_i is the multiplicative inverse of M/m_i in m_i.
Set $e_i = s_i \times M_i / m_i$

We get that $e_1 = 221$, $e_2 = -935$, $e_3 = 715$, and the unique solution modulo $11 \times 13 \times 17$ is 155. Finally, $S = 155 \equiv 2 \bmod 3$.

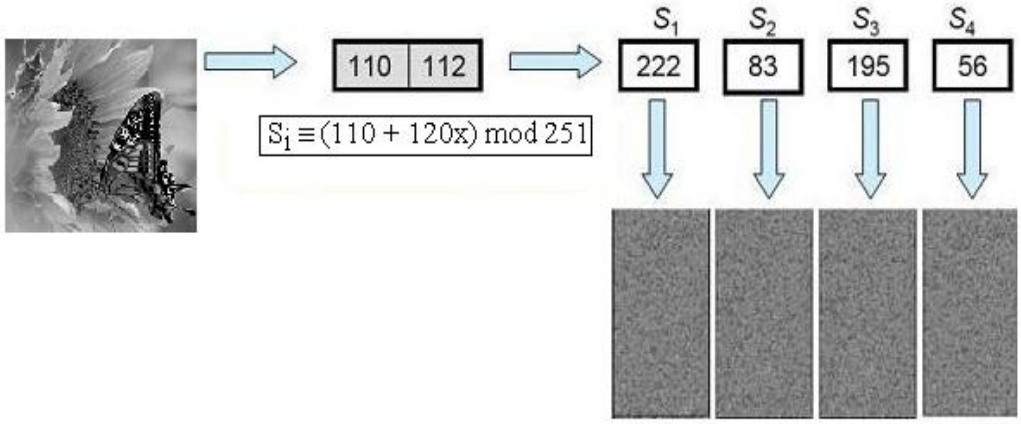
2.7.4 Thien & Lin's Secret Sharing Scheme

Thien and Lin [13] proposed a (k, n)-threshold-based Image Secret Sharing Scheme by cleverly using Shamir's SSS [1] to generate shares. The essential idea is to use a polynomial function of order (k-1) to construct n image shares from a $l \times l$ pixel secret image (denoted as I) as

$$S_x(i, j) = I(ik + 1, j) + I(ik + 2, j)\, x^1 + \text{---} + I(ik + k, j)\, x^{k-1} \pmod p$$
where $0 \le i \le \lfloor l / k \rfloor$ and $1 \le j \le l$

This method reduces the size of image shares to become 1/k of the size of the size of the secret image. Any k image shares are able to reconstruct every pixel value in the secret image.

Let us consider an example of (2, 4) image secret share construction process is illustrated in following figure where k=2 and n=4.



Thien & Lin's secret sharing for an Image

According to the technique, a first order polynomial function can be created as

Sx (i, j) = (110 + 112x) (mod 251)

Where 110 and 112 are the first two pixel values in the above considered image. For our four participants, we can randomly pick four x values, and substitute them into the polynomial function by setting p value to be 251 which is the largest prime number less than 255 (maximum gray image value). Four shares are computed as (1, 222), (2, 83), (3, 195) and (4, 56). They become the first pixel in four image shares. The second pixel is computed in the same manner by construction another first order polynomial function using next two pixels in the considered image. This process continues until all pixels are encoded. Four image shares are the bottom right images shown in figure above. And the size of four image shares are half (1/2) size of the original image. None of the image shares appear to reveal information about secret image. However, the pixel values in a natural image are not random because the neighboring pixels often have equal or close values. It is evident that the first two pixel values (110 and 112) are very close to each other. That creates the possibility that one image secret share may be used to recover the secret image by assuming the neighboring pixels have the same values in the first order polynomial function.

Since method reduces the size of image shares to become 1/k of the size of the secret image; the scheme cannot be qualified as a Perfect Secret Sharing Scheme (PSSS). In other words, the information about the secret exposed is proportional to the number of shares available until the number of shares becomes k or more. In addition, the field values in a natural image are not random because the neighboring pixels often have equal or close values. A secret image can be possibly recovered from less than k image shares because neighboring pixels are highly correlated.

To address these security issues, Thien & Lin suggested an idea by permutation the order of pixel (with permutation key) in the secret image before the image shares are computed. Conversely, the secret image can still be reconstructed from any k image shares by solving the permutated image and applying inverse-permutation using the permutation key.

Nevertheless, the permutation key becomes the single-point-failure in the system because the key can get lost or corrupted. This scheme also prevents real-time processing because the permuted image has to be obtained before the secret file can be reconstructed.

2.8 Steganography

Since the rise of the Internet one of the most important factors of information technology and communication has been the security of information. Cryptography was created as a technique for securing the secrecy of communication and many different methods have been developed to encrypt and decrypt data in order to keep the message secret. Unfortunately it is sometimes not enough to keep the contents of a message secret, it may also be necessary to keep the existence of the message secret. The technique used to implement this, is called steganography. Steganography is the art and science of invisible communication. This is accomplished through hiding information in other information, thus hiding the existence of the communicated information. The word steganography is derived from the Greek words "stegos" meaning "cover" and "grafia" meaning "writing" defining it as "covered writing". In image steganography the information is hidden exclusively in images.

The idea and practice of hiding information has a long history. In Histories the Greek historian Herodotus writes of a nobleman, Histaeus, who needed to communicate with his son-in-law in Greece. He shaved the head of one of his most trusted slaves and tattooed the message onto the slave's scalp. When the slave's hair grew back the slave was dispatched with the hidden message. In the Second World War the Microdot technique was developed by the Germans. Information, especially photographs, was reduced in size until it was the size of a typed period. Extremely difficult to detect, a normal cover message was sent over an insecure channel with one of the periods on the paper containing hidden information. Today steganography is mostly used on computers with digital data being the carriers and networks being the high speed delivery channels.

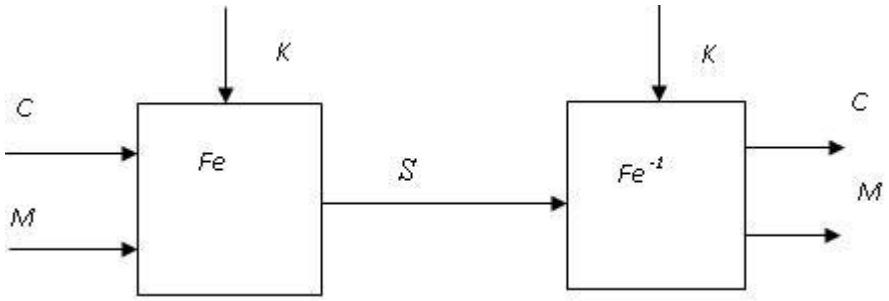
Steganography differs from cryptography in the sense that where cryptography focuses on keeping the contents of a message secret, steganography focuses on keeping the existence of a message secret. Steganography and cryptography are both ways to protect information from unwanted parties but neither technology alone is perfect and can be compromised. Once the presence of hidden information is revealed or even suspected, the purpose of steganography is partly defeated. The strength of steganography can thus be amplified by combining it with cryptography.

Modern steganography uses the opportunity of hiding information into digital multimedia files and also at the network packet level.

Hiding information into a media requires following elements

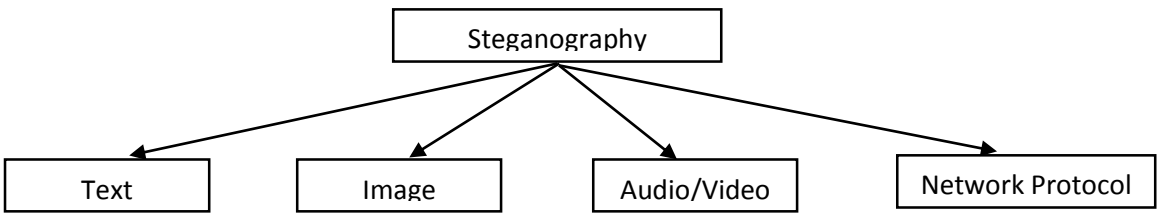
a. The cover media(C) that will hold the hidden data
b. The secret message (M), may be plain text, cipher text or any type of data
c. The stego function (Fe) and its inverse (Fe-1)
d. An optional stego-key (K) or password may be used to hide and unhide the message.

The stego function operates over cover media and the message (to be hidden) along with a stego-key (optionally) to produce a stego media (S). The schematic of steganography operation is shown below.



2.8.1 Review of Different Steganography Schemes

Almost all digital file formats can be used for steganography, but the formats that are more suitable are those with a high degree of redundancy. Redundancy can be defined as the bits of an object that provide accuracy far greater than necessary for the object's use and display. The redundant bits of an object are those bits that can be altered without the alteration being detected easily. Image and audio files especially comply with this requirement, while research has also uncovered other file formats that can be used for information hiding. Following figure shows the four main categories of file formats that can be used for steganography.



Hiding information in text is historically the most important method of steganography. An obvious method was to hide a secret message in every nth letter of every word of a text message. It is only since the beginning of the Internet and all the different digital file formats that is has decreased in importance. Text steganography using digital files is not used very often since text files have a very small amount of redundant data. Given the proliferation of digital images, especially on the Internet, and given the large amount of redundant bits present in the digital representation of an image, images are the most popular cover objects for steganography. This paper will focus on hiding information in images in the next sections. To hide information in audio files similar techniques are used as for image files. One different technique unique to audio steganography is masking, which exploits the properties of the human ear to hide information unnoticeably. A faint, but audible, sound becomes inaudible in the presence of another louder audible sound. This property creates a channel in which to hide information. Although nearly equal to images in steganography potential, the larger size of meaningful audio files makes them less popular to use than images. The term protocol steganography refers to the technique of embedding information within messages and network control protocols used in network transmission. In the layers of the OSI network model there exist covert channels where steganography can be used. An example of where information can be hidden is in the header of a TCP/IP packet in some fields that are either optional or are never used.

2.8.2 Text Steganography

In this technique the message is hidden within a plain text using different schemes like use of selected characters, extra white spaces of the cover text etc.

a. **Use of selected characters of cover Text:** Sender sends a series of integer number (Key) to the recipient with a prior agreement that the secret message is hidden within the respective position of subsequent words of the cover text. A “0” in the number series will indicate a blank space in the recovered message. The word in the received cover text will be skipped if the number of characters in that word is less than the respective number in the series (Key) which shall also be skipped during the process of message unhide.

b. **Use of extra white space characters of cover text**: A number of extra blank spaces are inserted between consecutive words of cover text. This numbers are mapped to a hidden message through an index of a lookup table. For example extra three spaces between adjacent words indicate the number "3" which subsequently indicates a specific text of a look-up table which is available to the both communicating parties as a prior agreement.

2.8.3 Image Steganography

The most widely used technique today is hiding of secret messages into a digital image. This steganography technique exploits the weakness of the human visual system (HVS). We cannot detect the variation in luminance of color vectors at higher frequency side of the visual spectrum. A picture can be represented by a collection of color pixels. The individual pixels can be represented by their optical characteristics like 'brightness', 'chroma' etc. Each of these characteristics can be digitally expressed in terms of 1s and 0s. E.g. A 24-bit bitmap will have 8 bits, representing each of the three color values (red, green, and blue) at each pixel. HVS can no detect any change while the LSBs of these color values are changed. In this scenario, as terminal recipients are human being, so this technique can be applied for image steganography.

a. **Image Steganography in Spatial domain:** In this method binary equivalent of the message (to be hidden) is distributed among the LSBs of each pixel color values. In this mechanism, no visual changes can be detected, only problem with this technique is that it is very vulnerable to attacks such as image compression and formatting.
b. **Image Steganography in Transformed domain:** Here images are first split into macro-blocks and apply Discrete Cosine Transform (DCT) on it. Then LSBs of binary equivalent the quantized DCT coefficient can be replaced by a bit from secret message. This way, message can be inserted on entire DCT coefficients of that image. This technique can very well be applied while creating an image in JPEG format, where each pixel of the image is digitally coded using discrete cosine transformation (DCT). The LSB of encoded DCT components can be used as the carriers of the hidden message

2.8.4 Audio/Video Steganography

In audio steganography, secret message is embedded into digitized audio signal which result slight altering of binary sequence of the corresponding audio file. There are several methods are available for audio steganography. Some of them are as follows

a. **LSB Coding:** Sampling technique followed by Quantization converts analog audio signal to digital binary sequence. In this technique LSB of binary sequence of each sample of digitized audio file is replaced with binary equivalent of secret message.
b. **Phase Coding:** Human Auditory System (HAS) can't recognize the phase change in audio signal as easy it can recognize noise in the signal. The phase coding method exploits this fact. This technique encodes the secret message bits as phase shifts in the phase spectrum of a digital signal, achieving an inaudible encoding in terms of signal-to- noise ratio.
c. **Spread Spectrum:** There are two approaches are used in this technique: the direct sequence spread spectrum (DSSS) and frequency hopping spread spectrum (FHSS). Direct-sequence spread spectrum (DSSS) is a modulation technique used in telecommunication. As with other spread spectrum technologies, the transmitted signal takes up more bandwidth than the information signal that is being modulated. Direct-sequence spread-spectrum transmissions multiply the data being transmitted by a pseudo random "noise" signal. In contrast, frequency-hopping spread spectrum pseudo-randomly retunes the carrier, instead of adding pseudo-random noise to the data, secret message can be used for the same.
d. **Echo Hiding:** In this method the secret message is embedded into cover audio signal as an echo. Three parameters of the echo of the cover signal namely amplitude, decay rate and offset from original signal are varied to represent encoded secret binary message. They are set below to the threshold of Human Auditory System (HAS) so that echo can't be easily resolved.

2.8.5 Network Protocol Steganography

This is another approach of steganography, which employs hiding data in the network datagram level in a TCP/IP based network over Internet. Network Covert Channel is the synonym of network steganography. Overall goal of this approach to make the stego datagram is undetectable by Network watchers like sniffer, Intrusion Detection System (IDS) etc. E.g. Information to be hide is placed in the IP header of a TCP/IP datagram. Some of the fields of IP header and TCP header in an IPv4 network are chosen for data hiding.

2.8.6 Steganalysis

Steganalysis is "the process of detecting steganography by looking at variances between bit patterns and unusually large file sizes". It is the art of discovering and rendering useless covert messages. The goal of steganalysis is to identify suspected information streams, determine whether or not they have hidden messages encoded into them, and, if possible, recover the hidden information. Unlike cryptanalysis, where it is evident that intercepted encrypted data contains a message. Steganalysis generally starts with several suspect information streams but uncertainty whether any of these contain hidden message. The steganalyst starts by reducing the set of suspect information streams to a subset of most likely altered information streams. This is usually done with statistical analysis using advanced statistics techniques.

2.8.7 Steganalysis Techniques

The properties of electronic media are being changed after hiding any object into that. This can result in the form of degradation in terms of quality or unusual characteristics of the media: Steganalysis techniques based on unusual pattern in the media or Visual Detection of the same.

For example in the case of Network Steganography unusual pattern is introduced in the TCP/IP packet header. If the packet analysis technique of Intrusion Detection System of a network is based on white list pattern (usual pattern), then this method of network steganography can be defeated.

In the case of Visual detection steganalysis technique a set of stego images are compared with original cover images and note the visible difference. Signature of the hidden message can be derived by comparing numerous images. Cropping or padding of image also is a visual clue of hidden message because some stego tool is cropping or padding blank spaces to fit the stego image into fixed size. Difference in file size between cover image and stego images, increase or decrease of unique colors in stego images can also be used in the Visual Detection steganalysis technique.

2.8.8 Steganography Attacks

Steganography attacks consist of detecting, extracting and destroying hidden object of the stego media. Steganography attack is followed by steganalysis. There are several types of attacks based on the information available for analysis. Some of them are as follows

a. **Known carrier attack:** The original cover media and stego media both are available for analysis.
b. **Steganography only attack:** In this type of attacks, only stego media is available for analysis.
c. **Known message attack:** The hidden message is known in this case.
d. **Known steganography attack:** The cover media, stego media as well as the steganography tool or algorithm, are known.

3 History behind the Development of Proposed Scheme

3.1 Preamble

We started our thesis work experimenting with various techniques of visual cryptography. The idea of this proposed multiple secret steganography scheme has come from some of those experiments done at the early stage of our thesis work. In this section, we shall discuss two of those secret sharing scheme, that helped us most to come up with this proposed scheme.

Visual Secret Sharing Scheme: we started with visual cryptography scheme, which is a new technique for providing information security that uses simple algorithm unlike the complex, computationally intensive algorithms used in traditional cryptography. This technique allows visual information to be encrypted in such a way that their decryption can be performed by the human visual system, without any complex cryptographic algorithms. This technique encrypts a secret image into a set of shares such that stacking a sufficient number of shares reveals the secret image. Shares are usually presented in transparencies. Even with the remarkable advance of computer technology, using a computer to decrypt secret is infeasible in some situations. For example, a security guard checks the badge of an employee or a secret agent recovers an urgent secret at some place where no electronic devices are applied. In these situations the human visual system is one of the most convenient and reliable tools to do checking and secret recovery.

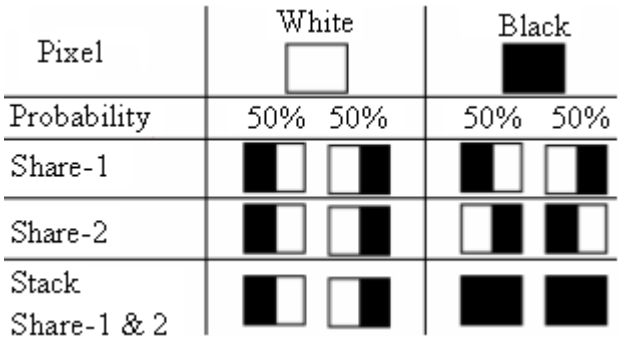
Visual Sharing Scheme for Embedding Multiple Images: In most of the known visual cryptography schemes, we have observed that each pixel in original image is encrypted using four (at least) pixels in the shared image and also four or more pixels are used for the visual decryption purpose. We know that if a pixel represented using N black & white (0 or 255 grey scale value) pixels, then it is possible to show N+1 distinct grey levels using the group of N black & white pixels. In existing visual sharing techniques, though four or more pixels are used for visual decryption but actually two different grey levels are used for visual image decoding. So, there is some redundancy in grey levels that kept unused.

Our proposed methodology actually tries address this redundancy issue and come up with a solution that utilizes this redundant grey levels and share more amount of information using the same amount of network bandwidth. Also this proposed methodology aims at sharing natural grey scale images. So this requires a conversion of natural grey scale image

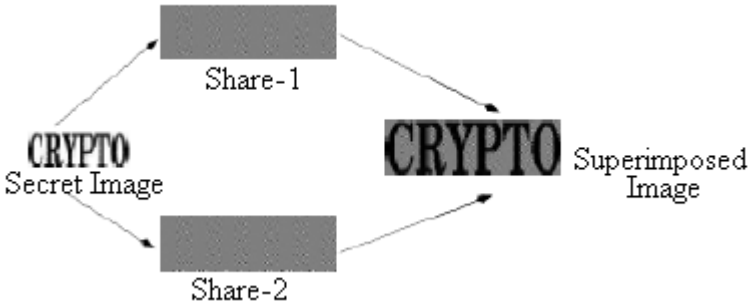
into corresponding half-tone image. Visually decrypted image quality heavily depends on image quality after half-tone image conversion. We shall discuss half toning techniques in details in the following sections of this chapter.

These two proposed scheme starts with the concept of Visual Cryptography that was first proposed by Naor and Shamir [15] assumes that the secret message consists of a collection of black and white pixels and each pixel is handled separately. Now suppose we want to distribute the secret image as shares among a set of two participants in such a way that from a single share it is not possible to reconstruct the secret image, but if both of the participants come together and stack their shares one above the other, the secret image will be visible with loss of contrast.

This scheme is known as the (2, 2)-Visual Threshold Scheme (in short (2, 2) –VTS). Naor and Shamir[15] devised the following scheme for the (2, 2)–VTS. The algorithm specifies how to share a single pixel, and it would be applied for every pixel in the image to be shared. A pixel P is split into two pixels in each of the two shares (each such pixel in the shares is called a sub pixel).



(2, 2)-VTS for black and white image



(2, 2)-VTS for an image

However, in this particular case the reconstructed pixel has gray level of 1 if P is black and a gray level of ½ if P is white. Thus, there is a 50% loss of contrast for white pixels. The

(2, 2)–VTS explained above can be mathematically represented by two 2 × 2 Boolean matrices S^0 and S^1 given as follows:

$$S^0 = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad S^1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

These two matrices are called basis matrices of the (2, 2)–VTS, where 1 stands for black pixel and 0 stands for white pixel.

Adhikari and Roy[16] extended the work of Naor and Shamir to a (2, n)-VTS, with n participants and pixel expansion m, using partially balanced incomplete block designs (PBIBD). Their model is defined by two n x m Boolean basis matrices S^1 and S^0, respectively for black and white pixels, such that

(a) $S^1_{\{i\}}$ and $S^0_{\{i\}}$ are equal up to a column permutation,

i.e., $w(S^1_{\{i\}}) = w(S^0_{\{i\}})$, $1 \le i \le n$;

(b) $w(S^1_{\{i,j\}}) > w(S^0_{\{i,j\}})$, $1 \le i < j \le n$.

The difference in the grey level of a black pixel and a white pixel in the recovered image determines the clarity of the recovered image. To measure this, the concept of relative contrast was first introduced in Naor and Shamir[15], who called it relative difference. Given a VCS with pixel expansion m, for any pair of participants i<j, (i, j=1,2,….,n), let

$$\alpha_{ij} = \frac{w(S^1_{ij}) - w(S^0_{ij})}{m}$$

be defined as the relative contrast for the recovery of an image by participants i and j for the VCS in another paper by Adhikari and Roy [17].

They explained the scheme using the PBIB (6, 4, 2, 3, 0, 1) by constructing a (2, 6)–VTS. The incidence matrix of this design is:

$$S^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad S^0 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

To share a black (white) pixel, we randomly permute the columns of S^1 (S^0) and allocate row i to participant i, j = 1, 2, …., 6. Thus, each participant gets as his share a 4-plet with

two 1's and two 0's in random order. Clearly, pixel expansion is 4 (m=4) and a participant cannot know whether his 4-plet came from a black or white pixel and thus security is guaranteed. This is done for each pixel in the original image as shown in following figure.

Pixel	Share	Share No.
□		1
		2
		3
		4
		5
		6
■		1
		2
		3
		4
		5
		6

(2, 6)-VTS for black and white image

Note: In this PBIB design, symbols 1 and 6 are first associates but 1 and 2 are second associates as mentioned in the following figure.

Pixel	Superposition of Shares 1 & 6	Superposition of Shares 1 & 2
□		
■		

Superposition of associates for (2, 6)-VTS

So, $\alpha 16 = ½$, whereas $\alpha 12 = ¼$. The joined image suffers from a loss of contrast and we get two types of relative contrasts depending upon the participants. Also, this work is limited only to black and white images. It cannot be applied to continuous tone color images.

In addition to the threshold scheme, Adhikari and Roy[16] also proposed a construction for General Access Structure. This is explained by a real life example: Suppose there is a locker in a bank and the key of the locker must be distributed among the customers and certain employees of the bank in such a way that the locker of some customer will be opened only when that customer and certain designated employees will contribute their keys.

As an example, let B be the boss and M1, M2, U1 and U2 be two managers and two users respectively of the bank. Any user with boss can open the locker. But, if the boss is absent, then a user can open the locker together with both the managers. This is explained in the following figure. To tackle this kind of situation, we need general access structure.

Image-1: The Password (Trial)

Image-2: Share for Boss (B)

Image-3: Share for M1

Image-4: Share for M2

Image-5: Share for U1

Image-6: Share for U2

Image7: Sh B + Sh U1

Image8: Sh B + Sh U2

Image-9: Sh M1 + Sh M2 + Sh U1

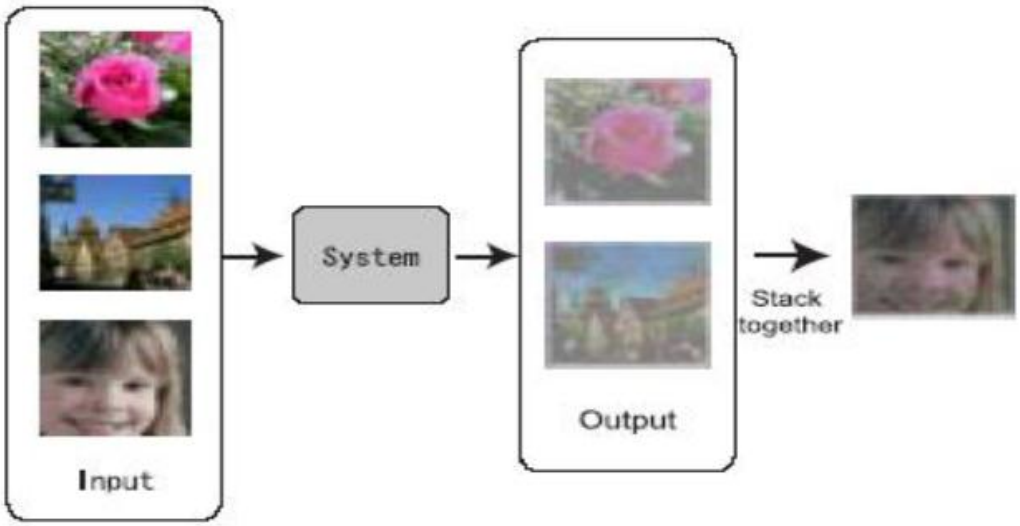
Image-10: Sh M1 + Sh M2 + Sh U2

Image-11: Sh M1 + Sh U1 + Sh U2

Image-12: Sh M1 + Sh M2 + Sh U1 + Sh U2

General Access Structure [Adhikari & Roy] for black and white Image

Another visual cryptography system proposed by Nakajima and Yamaguchi [18] takes 3 pictures as an input and generates 2 images which correspond to 2 of the 3 input pictures. The 3rd picture is reconstructed by printing the 2 output image onto transparencies and stacking them together. Following figure explains this scheme.



Basic idea of the Nakajima & Yamaguchi system

This type of visual cryptography, which reconstructs the image by stacking some meaningful images together, is especially called Extended Visual Cryptography [19]. There are some other researches that deal with color images. Naor and Shamir [20] discussed the visual cryptography scheme which reconstructs a message with two colors, by arranging the colored or transparent sub pixels.

Apart from above VSS Schemes, our scheme is equally applicable for any gray or color image. And reconstructed image is completely lossless. Here we generate shares using simple graphical ANDing with individual mask and reconstruction can be done by simple stacking the qualified set of shares without any computational operation.

3.2 Visual Secret Sharing Scheme

Here we are presenting stepwise protocol for our proposed Visual Secret Sharing (VSS) [22] [23] [24] scheme.

3.2.1 Sharing Phase

Input : A Secret Image, the value of n (total number of shares) & value of k (threshold).

Output: n Shared Images.

3.2.1.1 Algorithm

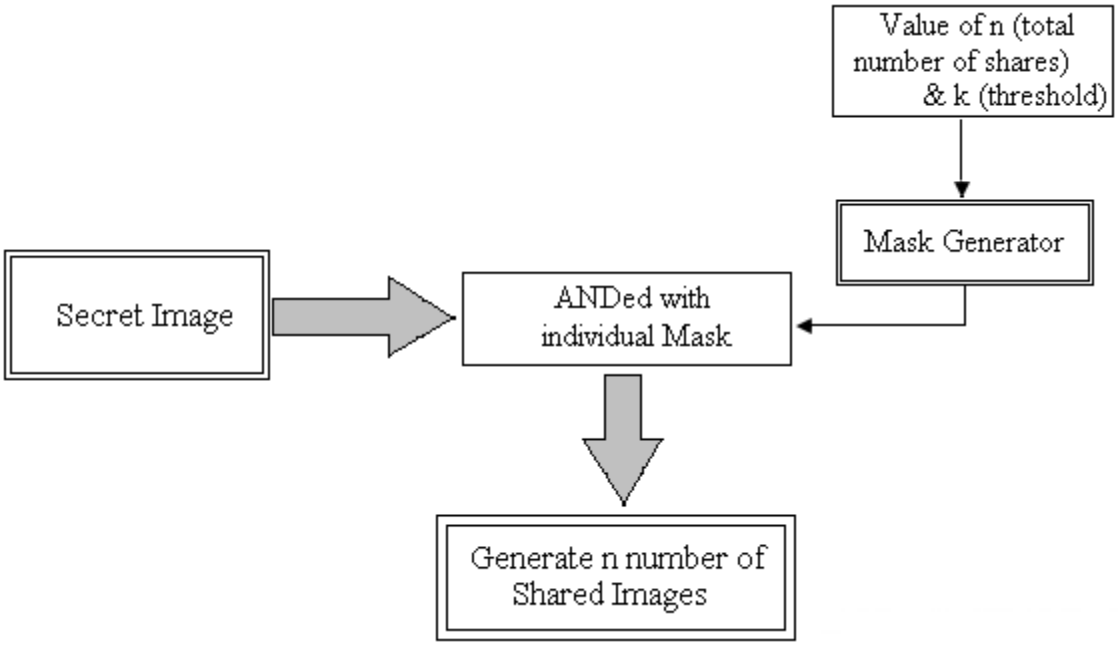
Step-1: Generate n masks using our mask generation algorithm [section-2.2.2] depending upon value of n (total number of shares) and k (threshold).

Step-2: Now the mask pattern is placed repeatedly on the entire Secret Image.

Step-3: The pixel corresponding to 1 in the mask pattern are kept as it is and pixel corresponding to 0 in the mask are replaced by white (for 24-bit image white pixel = (255, 255, 255)).

Step-4: By this way, we will get a single share. Same is repeated with all other mask patterns to form the other shares respectively.

3.2.1.2 Block diagram of sharing phase



Block diagram of our sharing phase (VSS)

3.2.2 Reconstruction Phase

Input : Collect k or more number of shared Image

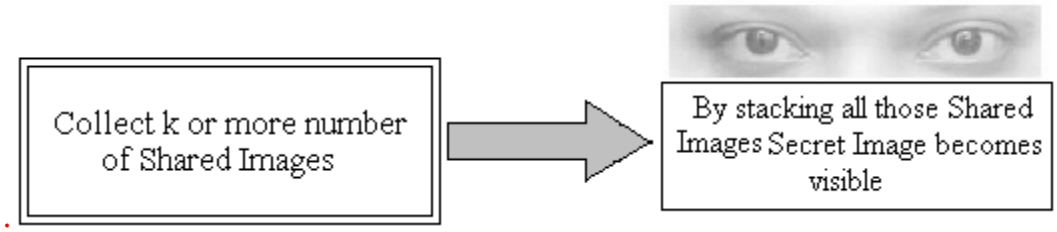
Output: Reconstruct Secret Image

3.2.2.1 Algorithm

Step-1: First collect any k or more shared images.

Step-2: By stacking all those shared images, the original secret image becomes visible, without any computational overhead.

3.2.2.2 Block diagram of reconstruction phase



Block diagram of our reconstruction phase (VSS)

3.3 Visual Sharing Scheme for Embedding Multiple Images

This proposed scheme starts with converting a grey scale image into a corresponding Half-tone image.

3.3.1 Converting Grey Scale to Halftone image

We have analyzed different half-toning algorithm and found error diffusion algorithms especially Floyd–Steinberg image dithering algorithm generates better quality halftone images from grey scale images.

The algorithm achieves dithering by diffusing the quantization error of a pixel to its neighboring pixels, according to the distribution

$$\frac{1}{16}\begin{bmatrix} & * & 7 \\ 3 & 5 & 1 \end{bmatrix}$$

The Element indicated with a star (*) indicates the pixel currently being scanned. The algorithm scans the image from left to right, top to bottom, quantizing pixel values one by one. Each time the quantization error is transferred to the neighboring pixels, while not affecting the pixels that already have been quantized. Hence, if a number of pixels have been rounded downwards, it becomes more likely that the next pixel is rounded upwards, such that on average, the quantization error is close to zero.

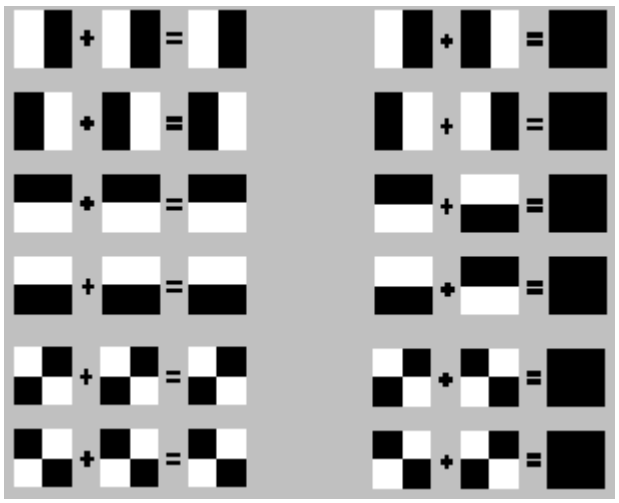
Code snippet of Floyd–Steinberg algorithm:

```
for(int i=0; i<ImageHeight; i++)
{
    for(int j=0; j<ImageWidth; j++)
    {
        pDstImageData[i*ImageHeight + j] = (pSrcImageData[i*ImageHeight + j] < 128) ? 0 : 255;
        int Error = pSrcImageData[i*ImageHeight + j] - pDstImageData[i*ImageHeight + j];

        if(j+1 < ImageWidth)
            pSrcImageData[i*ImageHeight + j +1] += (int)(((double)Error*7)/16);

        if(i+1 < ImageHeight && j-1 >= 0)
            pSrcImageData[(i+1)*ImageHeight + j -1] += (int)(((double)Error*3)/16);

        if(i+1 < ImageHeight)
            pSrcImageData[(i+1)*ImageHeight + j] += (int)(((double)Error*5)/16);

        if(i+1 < ImageHeight && j+1 < ImageWidth)
            pSrcImageData[(i+1)*ImageHeight + j + 1] += (int)(((double)Error)/16);
    }
}
```

3.3.2 Splitting Halftone Images

First step of Visual Secret Sharing is to split the halftone (converted from original image) image. We can split up the white and black pixels as specified in the following figure.

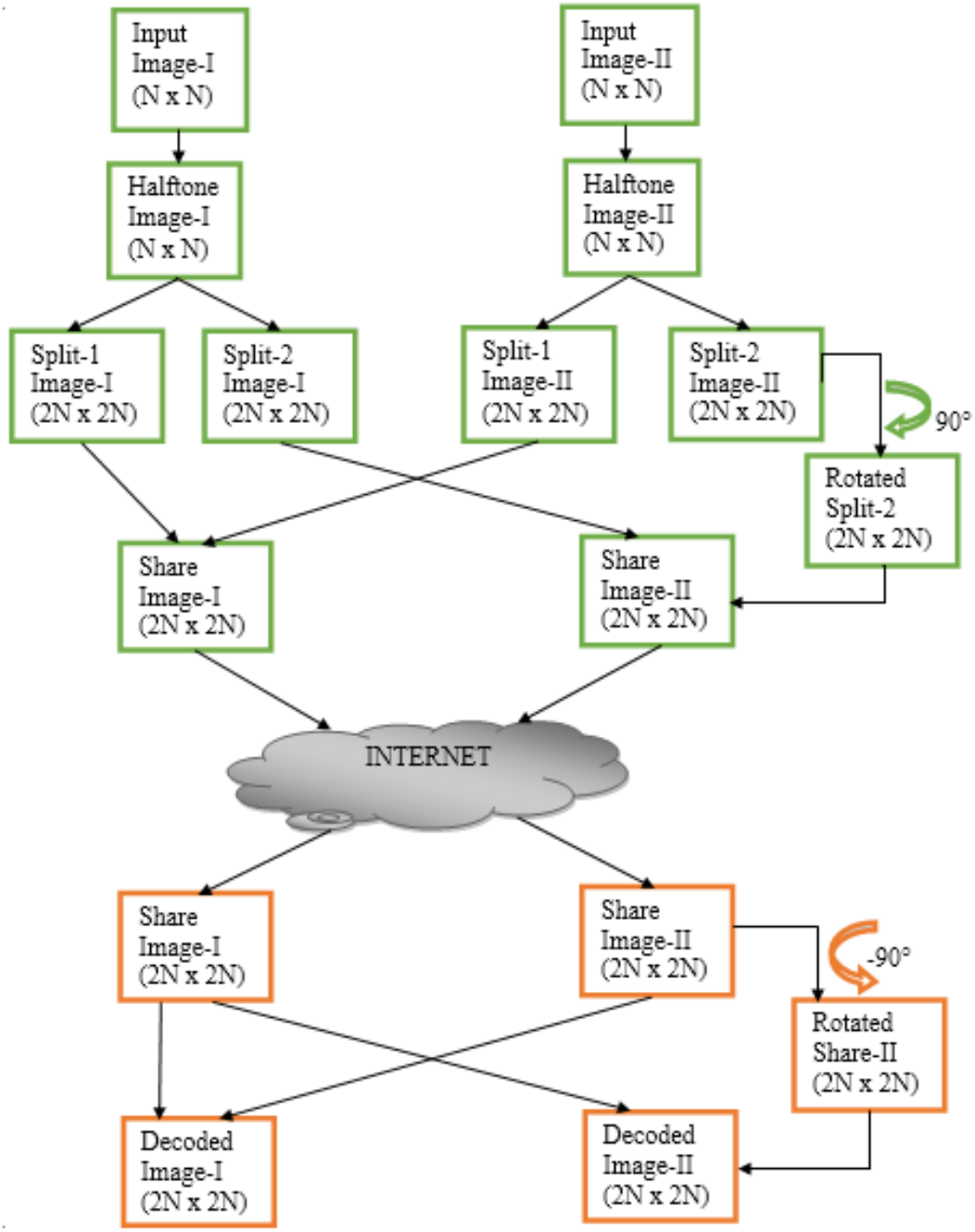
White Pixel Black Pixel



Here we represent a single pixel in actual halftone image using 4 pixels in the split image. Also it is observed from the above diagram that when split pixels corresponding to black are stacked together, then the output pixel become black. On the other hand, when split pixels corresponding to original white pixels are stacked together, output pixels are half black and half white. Eventually it appears as grey in output stacked image. This results

50% contrast loss for the white pixels. So the resultant stacked image may have overall contrast loss in the range of 0% (when all pixels are black in original image) to 50% (when all pixels are white in the original image).

3.3.3 Visual Sharing Scheme for two Images



Procedure for Image Encoding:

Step1: Convert original input grey scale images (N x N pixels, here 512 x 512 pixels) to corresponding halftone images (N x N pixels, here 512 x 512 pixels). Here Floyd-Steinberg method is used for halftone image conversion.

Step2: Now we need to Split both halftone converted images. White and black pixels are split in the following manner.

White pixel Black pixel

Here each pixel in halftone image is represented using four pixels in the split image. White pixels are split as [255, 0, 255, 0] and [255, 0, 255, 0] and black pixels are split as [255, 0, 255, 0] and [0, 255, 0, 255].

Step3: Rotate second split of the second image by 90° clockwise.

Step4: Stacked up 1st split of both the images and create 1st share image.

Step5: Stack up 2nd split of first image and rotated 2nd split of second image and generate the 2nd share image.

Procedure for Image Decoding:

Step1: Stack up (print both the shared image on transparent sheet and stack them up or stack them up programmatically) 1st and 2nd share image that will result visual decryption of 1st image.

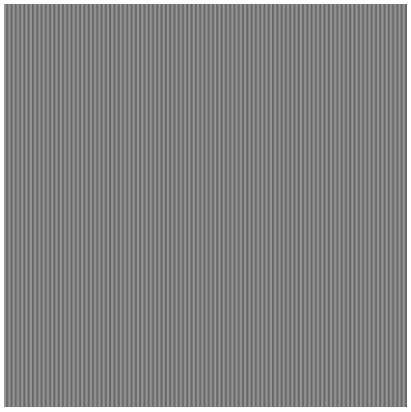
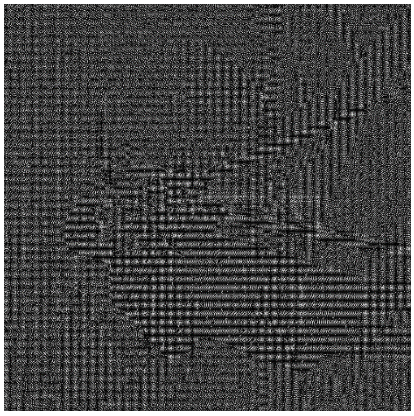
Step2: Rotate 2nd share image anticlockwise by 90°.

Step3: Stack up 1st share and rotated second shared image that will result visual decryption of 2nd image.

3.3.4 Experiment Results for Visual Sharing Scheme for two Images

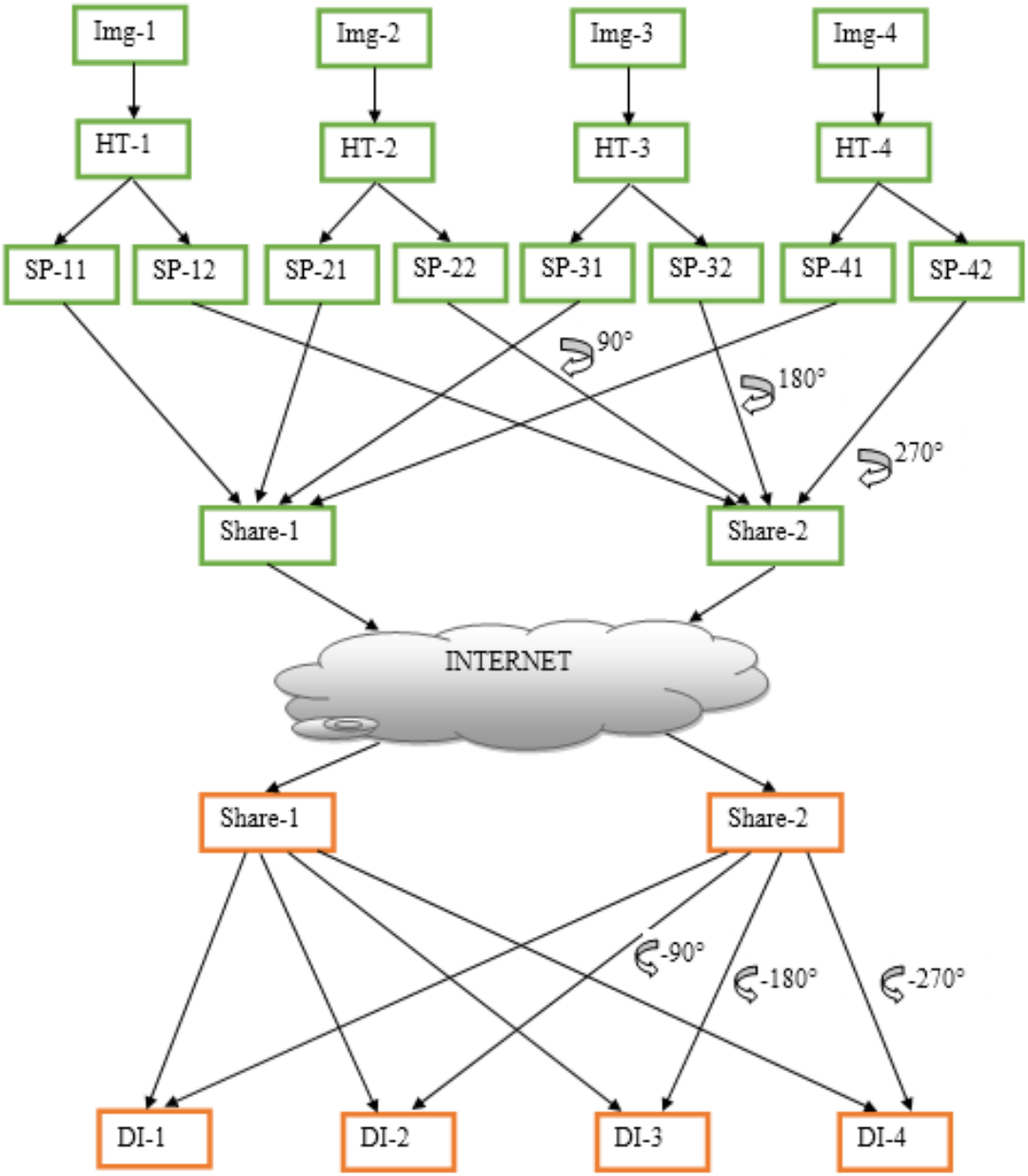
Input Images (512 x 512):


Encrypted Shared Images (1024 x 1024):

Visual Decrypted Images (1024 x 1024):


3.3.5 Visual Sharing Scheme for four Secret Images



3.4 Extension of the visual cryptography to shared Steganography

In our first visual cryptography scheme [22] [23] [24], we have used masking technique to implement threshold sharing. In the second visual sharing scheme, we have devised a technique to share multiple secrets using the same. Here we have combined as well as extended the above mentioned ideas to achieve this proposed Multiple Secret Steganography Scheme.

4 Proposed Scheme

4.1 Approach

Shamir[1], Blakley[2] and Asmuth-Bloom[3] Secret Sharing Schemes are briefly described in previous sections with their interesting features. Above schemes lead to high computational complexity during both sharing and reconstructing phase and most of the popular secret sharing schemes are based on above schemes. We have also describe Thien and Lin's[13] Secret Sharing Scheme in the previous section. The main drawbacks of Thien & Lin's scheme are single-point-failure for permutation key and also high computational complexity during both phases. Apart from those secret sharing technique, we are applying a scheme which deploys simple graphical masking, done by simple ANDing for share generation and reconstruction can be done by simple ORing the qualified set of shares. Not only that, it generates compressed shares that lead to strong protection of the secret file. Nevertheless it confirms confidentiality and integrity as well.

In this paper we have suggested a Multiple Secret Steganography Scheme based on novel Secret sharing scheme absolutely different from any of the schemes discussed so far, where simple graphical masking (ANDing) technique is used for intermediate share generation and then the shares are transmitted using envelope images using steganography techniques. On the other hand, reconstruction can be done simply ORing the predefined minimal set of intermediate shares once these shares are extracted out from envelope images based upon group key. The success of the scheme depends upon the mask generation, a step wise algorithm is suggested for such mask design for any (k, n)-threshold scheme where n number of masks are designed to generate n different shares and any k shares on ORing reconstruct the original secret.

The scheme ensures confidentiality and finally the integrity of the reconstructed secret file. Obviously one can use our proposed Multiple secret Steganography scheme with some simplified variants in many other similar applications. Now we shall discuss step by step about how we have built up this thesis work in the subsequent sections.

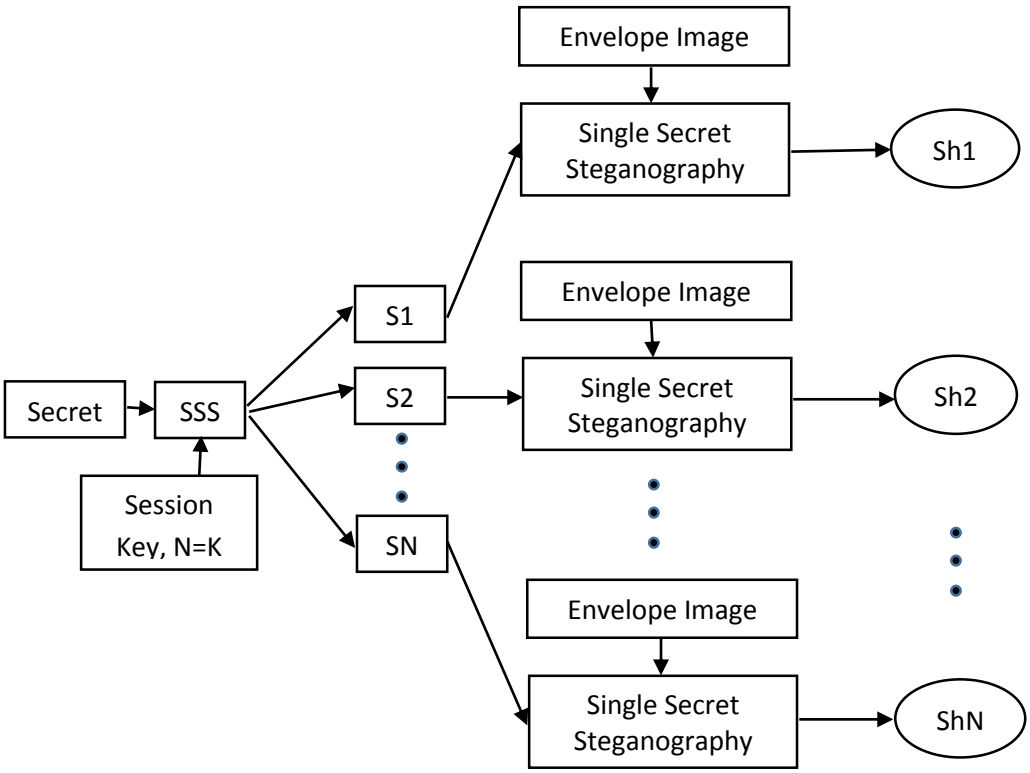
4.2 Single Secret Steganography with N=K

Objective of Multiple Secret Steganography Technique is to encrypt P number of secrets (S1, S2,…, Sp) that is shared among P different groups (G1, G2,…,Gp) and group specific data is decrypted based on group key. But as initial staring point of my Thesis work, I started working on sharing a single secret using Steganography Technique.

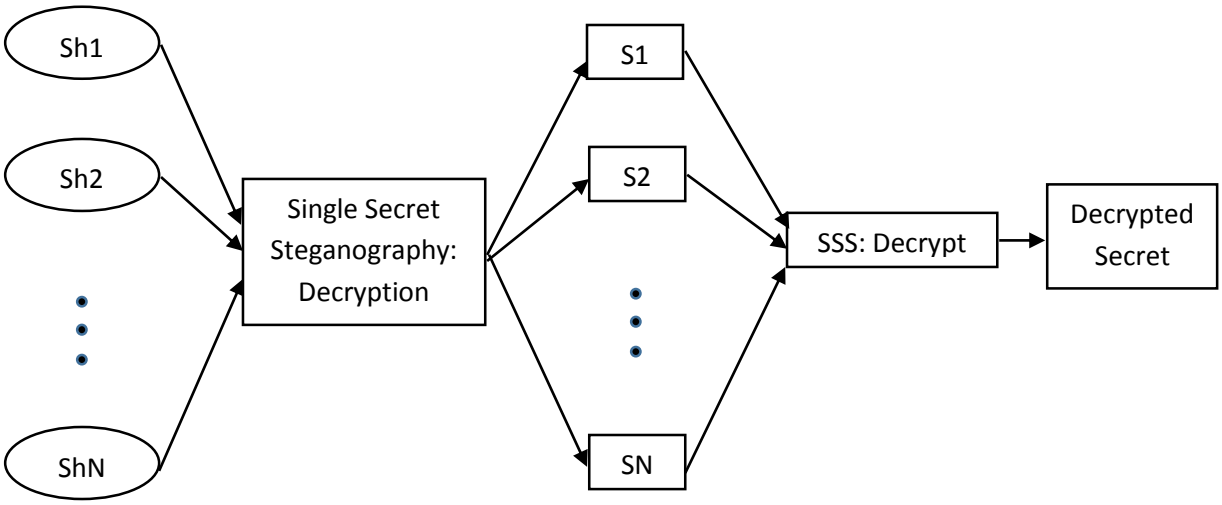
Here the objective is to share the secret data to a single group having N members in that group. In this Secret sharing scheme, first the secret data is encrypted and spitted {(S1, S2,…,SN)} into N shares. Then shares are taken one by one and encapsulated in envelope images. This process requires N envelope images as there are total N number of shares available. Here LSB of each envelope image is used to store the shares that is generated using Novel secret sharing Scheme (SSS). The reason we used LSB to ensure that there is no image quality degradation due to insertion of secret shares. In this fashion, all the N shares are encapsulated into N different envelope images. Now these N envelope images (encrypted data encapsulated within it) are final shares which are shared among N members of the group. Here the Threshold is kept as N (K=N) and group keys are not used intentionally to keep the initial implementation as simple as possible.

On the other hand, reconstruction or decryption of data requires all the N numbers of shares to be merged together. First step of decryption is to extract out the encrypted shares from the LSBs of shared images. To do so, length of encrypted shared data is passed as a part of shared envelope image. So the decryption process knows that exactly how much amount of data it requires to fetch to reconstruct the encrypted shared data. Once all the encrypted shared data got extracted out from all the envelope images, then SSS decryption methodology is used to decrypt the secret.

End to end data encryption and decryption technique is described using following figures.

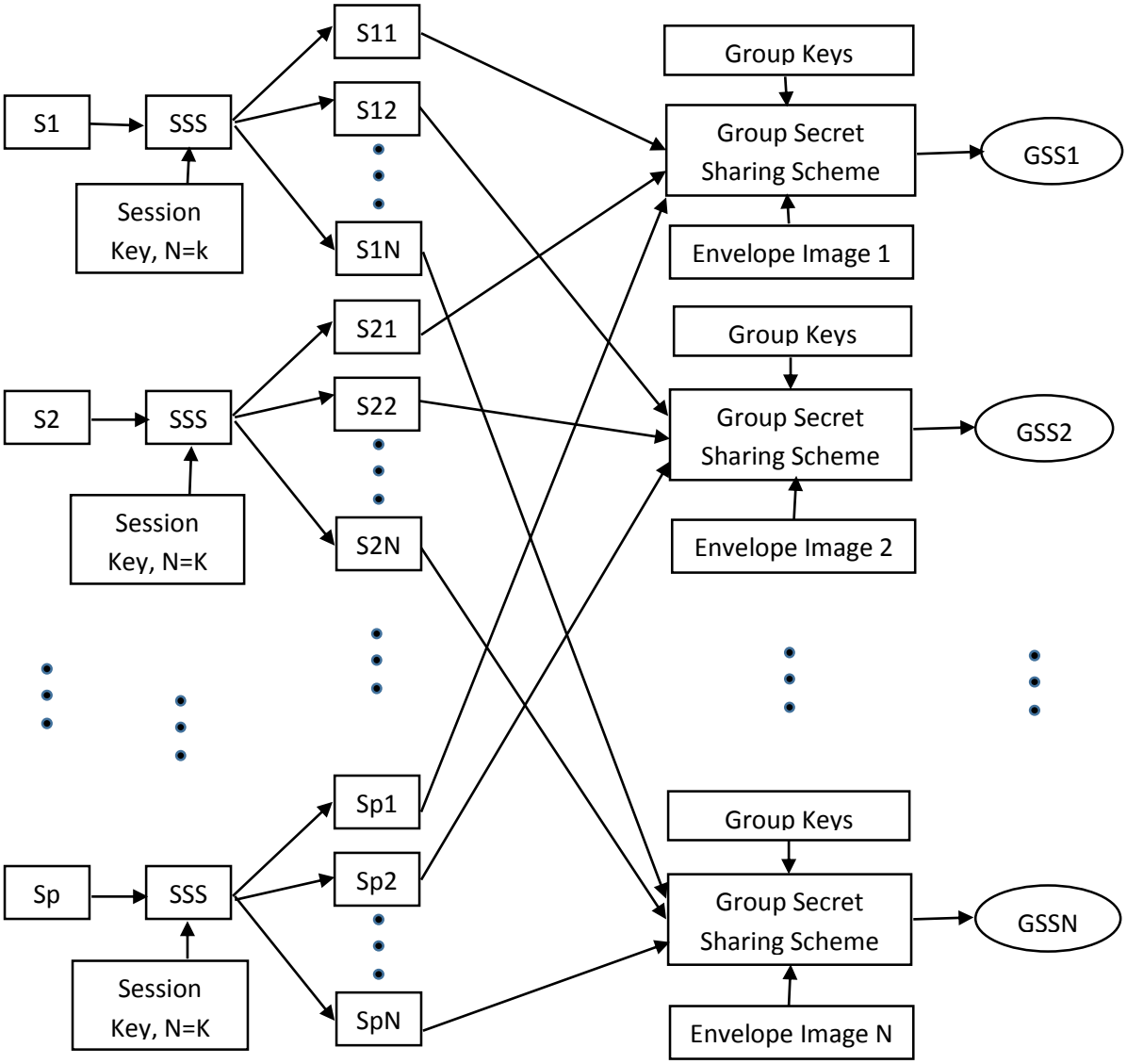


Single Secret Steganography: Encryption



Single Secret Steganography: Decryption

4.3 Multiple Secret Steganography with N=K and same values of N-K for each group



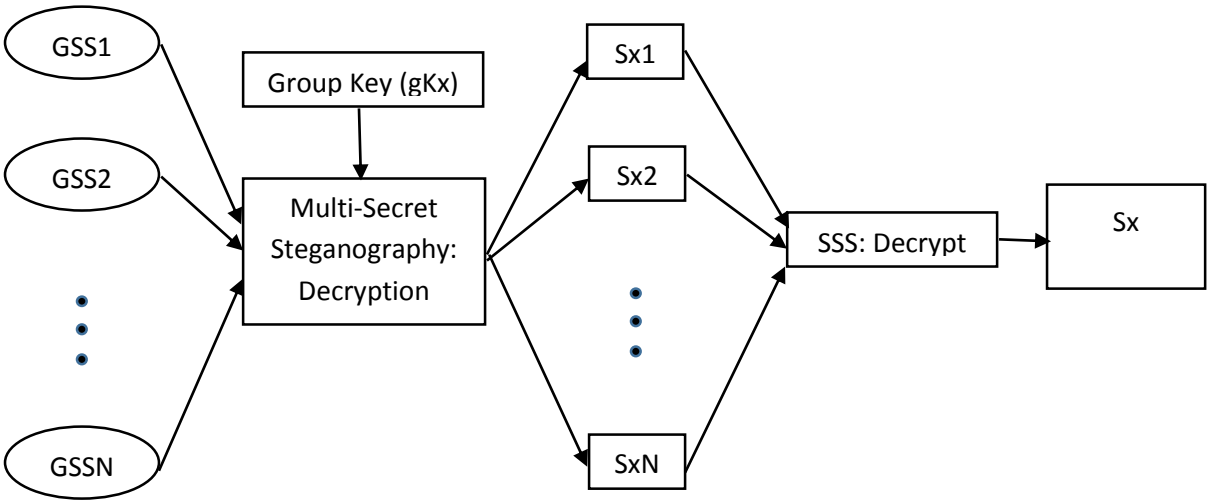
Multi-Secret steganography: Encryption (with N=K)

Once the single Secret Steganography technique, mentioned in the previous section, started working properly, I tried to apply the same methodology for multiple secrets. Instead of single secret, here multiple secrets are first broken into their respective shares using SSS algorithm and then they are inserted into corresponding envelope images. So each envelope image contains a partial fragment of shares from each secrets. Also I have introduced the concept of Group keys here. As multiple secrets and multiple groups are involved in this scheme, so we must require some group key to ensure group specific data security.

Here in this scheme, I have introduced some constraints to keep the implementation as simple as possible and at the same time I proceed one step towards desired goal. Here I have made assumptions that all the groups have same number (say N) of members that implies that total number of shares equal to N. Also all the N shares must require for reconstructing the secret for any group. This implies that Threshold vale K for each group is equal and its value is K=N.

The encryption process mentioned in the above flow diagram describes how the intermediate shares are generated from each secrets and then how each intermediate shares are inserted into envelope images. Here also LSBs are used to store shares into envelope images. LSB positions and color plane changes based on number of groups. Maximum number of groups supported for color envelope images is 9 and 3 for grey scale envelope images. I have restricted maximum 3 LSBs to ensure image quality. If there exists 9 groups {g1,g2,g3,g4,g5,g6,g7,g8,g9}, then the shared data is stored in envelope image in the following fashion {(Red, LSB_1), (Green,LSB_1), (Blue,LSB_1), (Red, LSB_2), (Green,LSB_2), (Blue,LSB_2), (Red, LSB_3), (Green,LSB_3), (Blue,LSB_3)}.

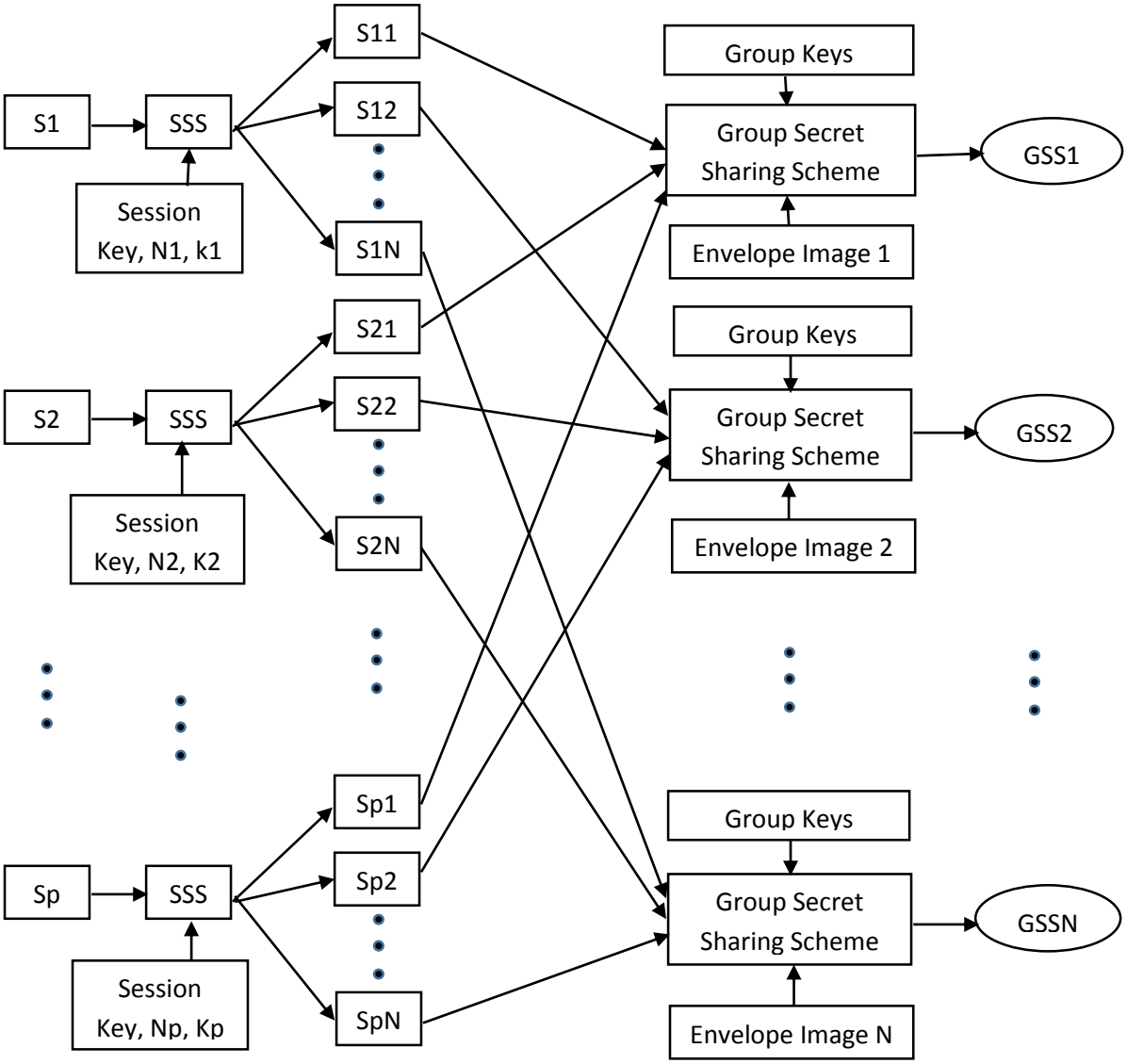
Apart from this, a data header has been added (detailed format of that data header is mentioned in the subsequent sections) while inserting the encrypted data into envelope image. This data header first encrypted using group key before inserting into envelope image. Information stored in the data header is required during decryption (decryption process mentioned in the following diagram). Data header mostly contains some information like color plane the data is stored, LSB position, random data start point etc. So group key is needed to decrypt group specific data header and that in turn required to decrypt the secret data.



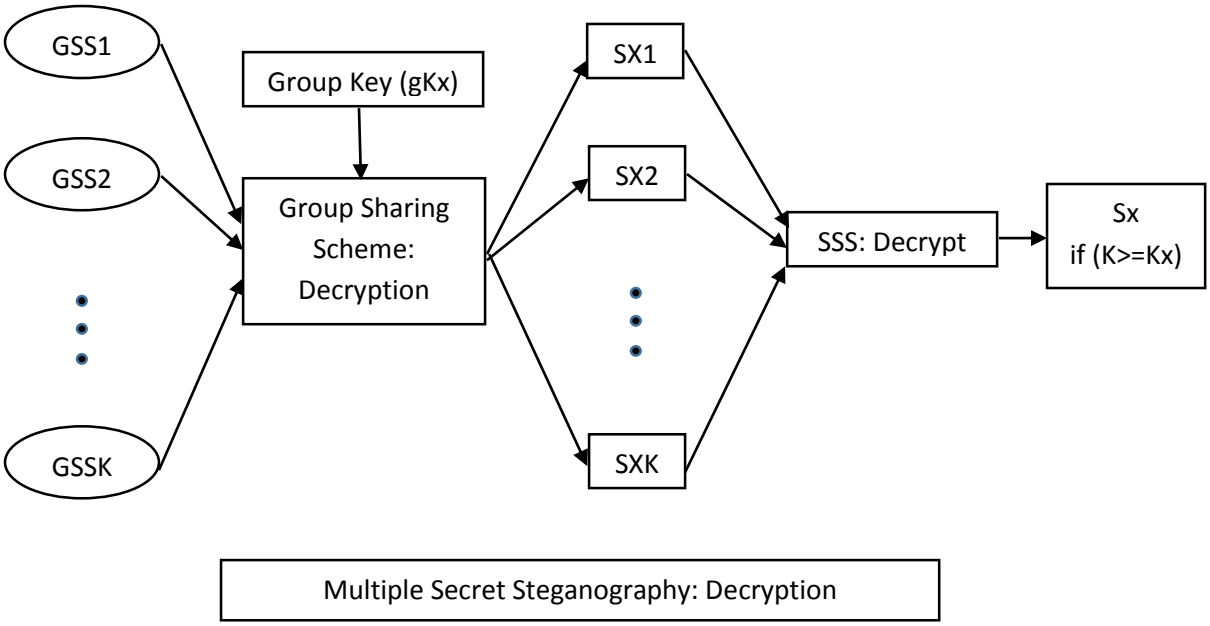
Multi-Secret steganography: Decryption (with N=K)

4.4 Multiple Secret Steganography with different N & K and N>=K for each group

This is the final step of implementation where I have introduced the concept of Threshold. Also added support for varying group members and threshold values for each group. All these points are discussed in details with the help of following diagrams.



Multiple Secret Steganography: Encryption



Here each group might have different number of group members, say N1, N2,……,Np (where p is the number of groups). Also Threshold (say K) for a particular group indicates that minimum those many numbers of shares are required to reconstruct the secret data for that specific group. Here the Threshold number for each group are K1,K2,…….,Kp.

If different groups have different number of group members (N1, N2, …., Np), then what should be the number of shares? Number of shares must corresponds to the highest number of group members among the p groups i.e. it should be equal to the highest value of N1, N2,……,Np.

So during encryption process, N (where N is the highest among N1,N2,…..,Np) number of intermediate shares are generated for each secret and also N number of final shares are generated using N envelope images. On the other hand, while decrypting a secret for a specified group, it requires threshold number (say K1 for group 1) of shares among N shares. So faithful reconstruction is ensured when number of available shares are more than or equal to the number of threshold value for that group.

4.5 Algorithm

The computation algorithm behind this proposed Multiple Secret Steganography scheme can be grossly divided into two parts.

(1) First part consists of logic for generating intermediate shares {(S11, S12, …,S1N), (S21, S22, …,S2N), …., (Sp1, Sp2, …,SpN)} from each secrets (S1, S2, …, Sp) intended for corresponding groups (assuming total number of groups equal to P). Here we used a novel secret sharing scheme [22] [23] [24] which employs simple graphical masking method using simple ANDing for share generation and reconstruction can be done by simple ORing the predefined minimal number (Threshold K) of shares.

(2) Second part consists of the logic for Multiple Secret Steganography to insert intermediate shares from each group into the corresponding envelope images i.e. for the first envelope image, all the first intermediate shares (S11, S21, …., Sp1) from each group are inserted. For second envelope image, all the second intermediate shares (S12, S22, …., Sp2) are inserted. Similar operation if done for all the N envelope images. These intermediate shares are inserted into LSBs of the envelope images for all the three different color planes (Red, Green, Blue). Also this secure insertion is done based upon Group keys, So while reconstruction, group key is the mandatory parameter for faithful reconstruction of secret for that particular group.

In the subsequent sections in this chapter, we shall be discussing the above mentioned algorithms in details.

4.5.1 Secret Sharing Scheme

For better understanding let us consider any secret as a binary bit file (i.e. it is the smallest unit to work upon, in actual implementation one can consider a byte or group of bytes or group of pixels as the working unit). The secret could be an image, an audio or text etc. We shall decompose the bit file of any size onto n shares in such a way that the original bit file can be reconstructed only ORing any k number of shares where $k \leq n \geq 2$ but in practice we should consider $2 \leq k < n \geq 3$.

Our basic idea is based on the fact that every share should have some bits missing and those missing bits will be replenished by exactly (k-1) other shares but not less than that. So every individual bit will be missed from exactly (k-1) shares and must be present in all remaining

(n-k+1) shares, thus the bit under consideration is available in any set of k shares but not guaranteed in less than k shares. Now for a group of bits, for a particular bit position, (k-1) number of shares should have the bit missed and (n-k+1) number of shares should have the bit present and similarly for different positions there should be different combinations of (k-1) shares having the bits missed and (n-k+1) number of shares having the bits present. Clearly for every bit position there should be ${}^{n}C_{k-1}$ such combinations and in our scheme thus forms the mask of size ${}^{n}C_{k-1,}$ which will be repeatedly ANDed over the secret in any regular order. Different masks will produce different shares from the secret. Thus 0 on the mask will eliminate the bit from the secret and 1 in the mask will retain the bit forming one share. Different masks having different 1 and 0 distributions will thus generate different shares.

Next just ORing any k number of shares we get the secret back but individual share having random numbers of 1's & 0's reflect no idea about the secret.

As an example a possible set of masks for 5 shares with threshold of 3 shares is shown below:

Share-1	:	1	1	1	0	1	1	0	1	0	0
Share-2	:	1	1	1	0	0	0	1	0	1	1
Share-3	:	1	0	0	1	1	1	0	0	1	1
Share-4	:	0	1	0	1	1	0	1	1	0	1
Share-5	:	0	0	1	1	0	1	1	1	1	0

One can easily check that ORing any three or more shares we get all 1's but with less than three shares some positions still have 0's i.e. remain missing.

4.5.1.1 The Mask Generation Algorithm

Here we are presenting the algorithm for designing the masks for n shares with threshold value k.

Step-1: List all row vectors of size n having the combination of (k-1) numbers of 0's and (n-k+1) numbers of 1's and arrange them in the form of a matrix. Thus dimension of the matrix will be ${}^{n}C_{k-1}\times$ n.

Step-2: Transpose the matrix generated in Step-1. Obvious dimension of the transposed matrix will be $n \times {}^{n}C_{k-1}$. Each row of this matrix will be the individual mask for n different shares. The size of each mask is ${}^{n}C_{k-1}$ bits, i.e. the size of the mask varies with the value of n and k.

4.5.1.2 Pseudo Code for mask generation

Input: n, k

Output: mask[n][ ] and length of mask pattern (say len)

```
int mask_generator(n, k, mask[n][])
{
    bin [][n] : integer array
    len = 0; //initialization
    max_val = 2^n – 1;
    //calculate the decimal value of n numbers of 1's

    For  i=max_val-1  to 0
          decimal_to_binary(i, bin[len][n]);
          //calculate binary equivalent of decimal i and store in bin [][] array
        If (zero_check(bin [ len ][ n ], k))
           len++;
         //check whether (k-1) nos. of zero exist or not, if true then increment len by 1
        end if.
      end for
      rearrange(bin); // rearrange the row of bin[][n] array
      transpose(mask, bin);  //take transpose matrix of bin [][n] and store in  mask[n][]
    return len;
}
```

Let us consider the previous example where n=5 and k=3.

Step-1: List of row vectors of size 5 bits with 2 numbers of 0's and 3 numbers of 1's.

1	1	1	0	0
1	1	0	1	0
1	1	0	0	1
0	0	1	1	1
1	0	1	1	0
1	0	1	0	1
0	1	0	1	1
1	0	0	1	1
0	1	1	0	1
0	1	1	1	0

Dimension of the matrix is ${}^5C_2 \times 5$
I.e. 10×5

Step-2: Take the transpose of the above matrix and we get the desired masks for five shares as listed above in the form of matrix of dimension $5 \times {}^5C_2$ i.e. 5×10. There are five masks each of size 10 bits

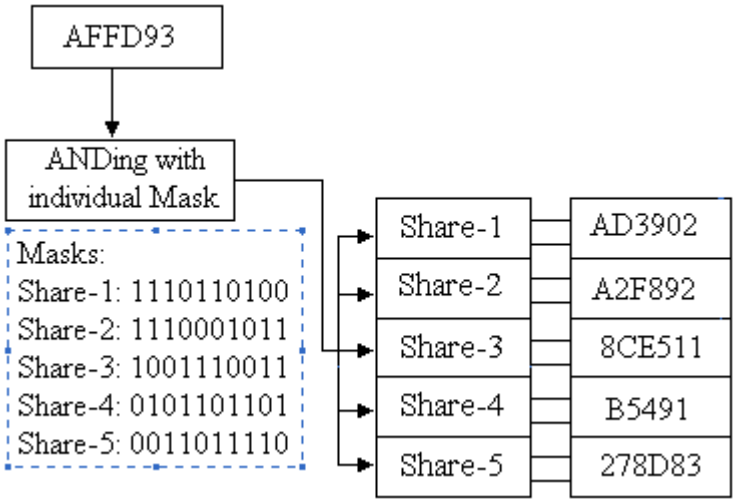
.This algorithm can be applied for any type of file because any file looked upon as a binary file. Here we have applied our scheme on three very frequently used file types namely, Image (.bmp file), Audio (.wav file) and Text (.txt file).

3.5.1.3 Example to describe our scheme

Let us consider (3, 5)-threshold scheme and the secret key is AFFD93.

Sharing Phase:

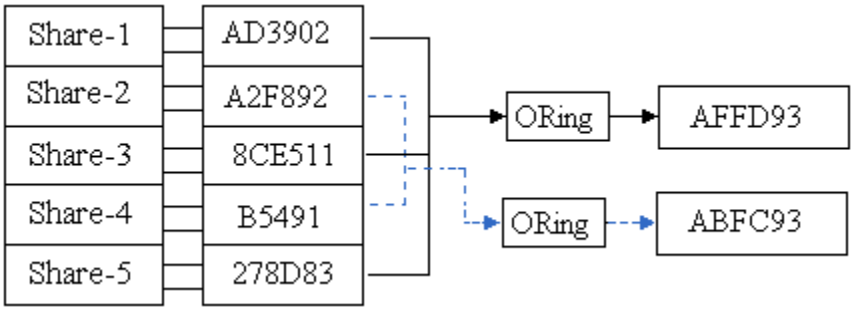
In sharing phase 5 shares value will be generated by simple bit wise ANDing operation with individual mask.



Key sharing using our proposed Secret sharing scheme

Reconstruction Phase:

In this reconstruction phase, if we collect any three or more shared value, we can easily reconstruct the secret key by applying simple bit wish ORing operation. Suppose we take share-1, 3, 5 values AD3902, 8CE511 and 278D83 respectively and apply bitwise ORing operation that easily reconstructs the secret AFFD93. But less than 3 shares cannot reconstruct the secret key.



Key reconstruction using our proposed Secret sharing scheme

4.5.2 Multiple Secret Steganography

In this proposed multiple secret steganography technique, we have used Image Steganography to hide the encrypted secret messages. As per this proposed steganography scheme broadly mentioned in the previous sections, if we have N number of intermediate shares from each group, then we require total N number of cover images to hide those intermediate shares. To explain this steganography algorithm in details, we can start with how the first intermediate shares (S11, S21, …, SN1) from each group are inserted into first envelope image. To insert each intermediate share data into the cover image, we generate data header for each of the shares. The data header consists of some parameters mentioned in the following diagram.

Group Key	Data length (in bytes)	Random Start offset	Color Plane	LSB position

Group Key: This key is specific to each group and it is available to the each member of that group. This group key is a must requirement for decryption of secret data. This group key is shared as part of data header serve as group identifier.

Data length: Length of encrypted intermediate shares in bytes.

Random Start offset: This is a positive random number generated by a random() function. This number ranges 0-255 and it indicates the start pixel position from where intermediate shared data are stored for a specified color plane and LSB position.

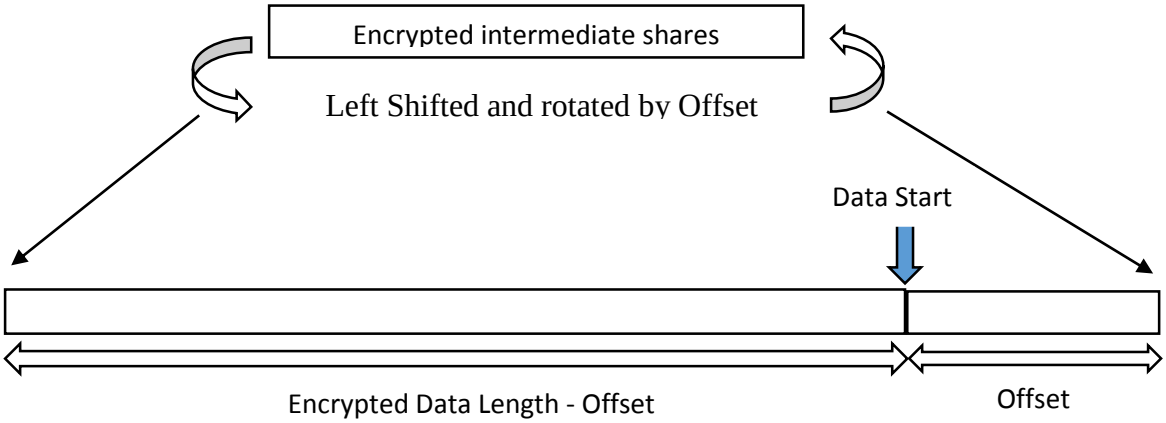
Color Plane: Indicate in which color plane secret data is stored.

LSB Position: Indicates in which LSB position for a specified color plane, the secret data is stored.

Once the header is generated with all the above mentioned information, the header data is encrypted using the group key (say gK1 is the group key for the first group) for the corresponding group. This ensures a secure header transmission through steganography image. With this encrypted data header, actual intermediate share is stored in the following fashion.

Header Length [4 bytes long]	Encrypted data Header [124 bytes long]	Data Start position is shifted by Random Offset	Encrypted intermediate shares

To minimize the possibilities of any unauthorized access, the encrypted intermediate share data is left shifted and rotated by the amount of offset mentioned in the data header.



In this way, final data format is generated along with header information. Now we shall discuss the logic for inserting these data stream into cover image.

Now we shall look into how color plane and LSB positions are selected for an intermediate share data stream. If we assume that we have 3 different secrets for 3 different groups, then the distribution of data for first cover image is as follows.

Group Number	Intermediate Share ID	Color Plane	LSB Position
1	S11	Red	1
2	S21	Green	1
3	S31	Blue	1

If we assume that we have 5 different secrets for 5 different groups, then the distribution of data for first cover image is as follows.

Group Number	Intermediate Share ID	Color Plane	LSB Position
1	S11	Red	1
2	S21	Green	1
3	S31	Blue	1
4	S41	Red	2
5	S51	Green	2

As per this present steganography scheme, Maximum 9 groups can be supported when we use 24-bit color cover image. For 8-bit grey scale cover image, secrets for maximum 3 groups can be shared. This limitation is due to the fact that we are allowing maximum 3 LSB positions of a byte of cover image pixel data can be used to hide the secret messages. This ensures that there is no visually perceived distortion in the cover image due to insertion of secret messages. If we assume that we have Maximum 9 different secrets for 9 different groups, then the distribution of data for first cover image is as mentioned in the following table.

Group Number	Intermediate Share ID	Color Plane	LSB Position
1	S11	Red	1
2	S21	Green	1
3	S31	Blue	1
4	S41	Red	2

5	S51	Green	2
6	S61	Blue	2
7	S71	Red	3
8	S81	Green	3
9	S91	Blue	3

Till now we have seen how secret messages are inserted into the first cover image. This same procedure is followed to insert messages for all the cover images (24-bit color). If we have 8-bit grey scale cover images, then we can share Maximum 3 different secrets for 3 groups and the distribution of data for first cover image is as follows.

Group Number	**Intermediate Share ID**	**Color Plane**	**LSB Position**
1	S11	Grey	1
2	S21	Grey	2
3	S31	Grey	3

Algorithm for insertion of intermediate shares into cover image is given below.

```
public void InsertSecretData(List<SecretData> SecretDataList)
{
    Dictionary<string, int> ColorMap = new Dictionary<string, int>();
    ColorMap.Add("R", 0);
    ColorMap.Add("G", 1);
    ColorMap.Add("B", 2);

    int dataLength = SecretDataList.Count;

    for (int iY = 0; iY < BmpImageData.Height; iY++)
    {
        for(int iX = 0; iX < BmpImageData.Width; iX++)
        {
            Color pixelData = BmpImageData.GetPixel(iX, iY);
            byte [] pixelRGB = new byte[]{pixelData.R, pixelData.G, pixelData.B};
            int bitPos = 0;
            for (int index = 0; index < dataLength; index++)
            {
                bitPos = GroupIDs.getBitPosition(index);
                if (SecretDataList[index].IsDataAvailable())
                {
                    if (SecretDataList[index].
                              GetDataBit( GroupIDs.getDataOffset(index)))
                    {
                        pixelRGB[ColorMap[GroupIDs.getColorPosition(index)]]
                                  |= (byte)((UInt32)0x1 << bitPos);
                    }
                    else
                    {
                        pixelRGB[ColorMap[GroupIDs.getColorPosition(index)]]
                                  &= (byte)~((UInt32)0x1 << bitPos);
                    }
                }
            }
            Color newPixelData = Color.FromArgb(pixelRGB[0],pixelRGB[1],pixelRGB[2]);
            BmpImageData.SetPixel(iX, iY, newPixelData);
        }
    }
}
```

How many numbers of intermediate shares would be generated from each secret data, that is how many number of cover images are required is depends on number of members available in the groups. Suppose there exists P groups and each group has N1, N2, …., Np number of members respectively, then required number of cover images N is equal to highest value among N1, N2, …, Np. Encrypted intermediate secrets for P groups are inserted into these N envelope images to generate N shares. From these N shares N1 number of copies distributed to the 1st group members, N2 copies are distributed to 2nd group members, and so on for rest of the groups.

Now we shall discuss about decryption methodologies. Here decryption implies reconstruction of a secret for a specific group. Each group is identified by its corresponding group keys (say group key for group1 is gK1). So group key is the must requirement for decrypt group specific secrets.

Also there exists a threshold value (K1, K2, …, Kp) for each group. Threshold value for a group (say K1 for the first group) indicates the minimum number of shares that needs to be combined to reconstruct the secret data. If we assume that every group member has one share with them, this indicates when threshold number of group members combines their shares, then only they can extract meaningful information from the Stego Images.

Once we have threshold number of stego images for a group and its group key, we can initiate the decryption process. Decryption process starts with finding out matched header for the specified group key from each of the stego images. For each 24-bit color stego images, all possible 9 secret data channels {(Red,LSB_1), (Green,LSB_1), (Blue,LSB_1), (Red,LSB_2), (Green,LSB_2), (Blue,LSB_2), (Red,LSB_3), (Green,LSB_3), (Blue,LSB_3)} are read and 132 bytes (4 bytes header length and 128 bytes of encrypted header data) data from each channel are extracted out and populated in an array of header string of length 9. Now actual encrypted header data is extracted out for each header based on length of the header. Any length greater than 128 bytes (max encrypted header length) indicates malformed header and hence discarded. Now we execute header decryption methods for each header one by one using the specified group key. Header decryption operation throws exception for an unmatched header. This header decryption operation continues until it found a matched header. In case any improper group key used for decryption, it will never found a matched header and hence decryption operation fails.

Algorithm for extracting out headers from steganography image is mentioned below.

```
for (int iY = 0; iY < ImageData.Height; iY++)
{
    for (int iX = 0; iX < ImageData.Width; iX++)
    {
        if ((iY * ImageData.Width + iX) < Header_Length*8)
        {
            Color pixelData = ImageData.GetPixel(iX, iY);

            for (int ChIndex = 0; ChIndex < MaxDataChannels; ChIndex++)
            {
                int LSBPos = 0;
                if (ChIndex < 3)
                {
                    LSBPos = 0;
                }
                else if (ChIndex >= 3 && ChIndex < 6)
                {
                    LSBPos = 1;
                }
                else if (ChIndex >= 6 && ChIndex < 9)
                {
                    LSBPos = 2;
                }
                UInt32 MessageData = 0;
                UInt32 ColorPlane = 0;
                if ((ChIndex%3)==0)
                {
                    ColorPlane = pixelData.R;
                }
                else if (((ChIndex+2) % 3) == 0)
                {
                    ColorPlane = pixelData.G;
                }
                else if (((ChIndex + 1) % 3) == 0)
                {
                    ColorPlane = pixelData.B;
                }
                MessageData = ColorPlane & ((UInt32)0x1 << LSBPos);
                if (!(HeaderList[ChIndex].IsHeaderDataCompleted()))
                {
                    HeaderList[ChIndex].SetHeaderBit(MessageData > 0
                                                        ? true : false);
                }
            }
        }
        else
        {
            break;
        }
    }
}
```

In this algorithm, we iterate pixel by pixel over the steganography image to extract out all the header information for all the 9 channels. We continue this iteration for (Header_Length*8) times i.e. (128*8) times as header length is 128 bytes. Each pixel data has three components R, G, B i.e. Red, Green, Blue. We extract out 128 bytes of header data for each of the color planes for 3 LSB positions. This way, we form the header list for all the 9 possible headers and then we search for a matched header corresponding to the group key in the list of 9 headers. Algorithm for finding out matched data header is given as follows.

```
int FoundMatchedIndex = -1;
for (int i = 0; i < MaxNumofHeaders; i++)
{
    DecryptedHeader = HeaderList[i].FindMatchedGroupHeader(groupKey);
    if (DecryptedHeader != null && DecryptedHeader.length > 0)
    {
        FoundMatchedIndex = i;
        break;
    }
}

public string FindMatchedGroupHeader(string groupKey)
{
    string MatchedHeader = null;

    byte[] LenBytes = AllocateMemory[4];
    for (int i = 0; i < 4; i++)
        LenBytes[i] = HeaderData[i];

    int Length = ConvertToInt(LenBytes);
    byte[] EncrypetdDataHeader = AllocateMemory[Len];
    for (int i = 0; i < Length; i++)
        ActualDataHeader[i] = HeaderData[i+4];

    string DecryptedHeader = StringCipher.Decrypt(EncrypetdDataHeader, GroupKey);
    if (null == DecryptedHeader)
        return null;

    string ExtractedGroupKey = DecryptedHeader.GetBytes(16);
    if (ExtractedGroupKey.Equals(GroupKey))
    {
        MatchedHeader = DecryptedHeader.Remove(0, 16);
    }
    return MatchedHeader;
}
```

As I already mentioned earlier that every encrypted data header also contains the group key in the first header field and it is used as group identifier. Thus the matched header is identified by comparing the group key with the first header field. Once the matched header found, then rest of the header elements (i.e. Data length, Data start offset, color plane and LSB position) are parsed using the following algorithm.

```
private bool ParseDataHeaderElements(string DataHeader)
{
    string[] SplitGroupId = DataHeader.Split('_');
    if(SplitGroupId != null && SplitGroupId.size == 4)
    {
        dataLength_ = ConvertToInt(SplitGroupId[0]);
        dataOffset_ = ConvertToInt(SplitGroupId[1]);

        if (SplitGroupId[2].Equals("grey"))
        {
            colorType_ = "grey";
            colorPosition_ = "grey";
        }
        else
        {
            colorType_ = "color";
            colorPosition_ = SplitGroupId[2];
        }

        LSBPosition_ = ConvertToInt(SplitGroupId[3]);
        return true;
    }
    else
    {
        return false;
    }
}
```

Once we parse the data header, we have revealed all the secret information that is required to extract out all the secret data. Now we have information that which color plane (whether R, G or B) and which LSB position, secret data is stored. Also we know the offset which indicates that starting from which pixel location, the encrypted intermediate shares are stored. Another information we have is data length in bytes, which indicates how many bits we need to extract so that no portion of encrypted data got left out. Algorithm for extraction of intermediate encrypted shares are mentioned below.

```
int colorPlaneVal = colorPlaneMap[colorPosition_];

for (int iY = 0; iY < ImageData.Height; iY++)
{
    for (int iX = 0; iX < ImageData.Width; iX++)
    {
        if ((iY * ImageData.Width + iX) >= (Header_Length*8) + dataOffset_)
        {
            Color pixelData = ImageData.GetPixel(iX, iY);

            UInt32 ColorPlaneData = 0;
            if (colorPosVal == 1)
            {
                ColorPlaneData = pixelData.R;
            }
            else if (colorPosVal == 2)
            {
                ColorPlaneData = pixelData.G;
            }
            else if (colorPosVal == 3)
            {
                ColorPlaneData = pixelData.B;
            }
            UInt32 MessageData = ColorPlaneData & ((UInt32)0x1 << LSBPosition_);

            if (!SecretData.IsAllDataFound())
            {
                SecretData.SetDataBit(MessageData > 0 ? true : false);
            }
        }
    }
}
```

In this way, encrypted intermediate share is extracted out for the specific group identified by group key. This way, encrypted intermediate shares are extracted out for all the available steganography images. Once all the intermediate shares got extracted out, we apply Secret Sharing Scheme (SSS) decryption methodology to reconstruct actual secret for the specified group. Here number of available intermediate shares i.e. number of available steganography images must be equal to or greater than the threshold number for that specified group, otherwise secret message reconstruction is not possible.

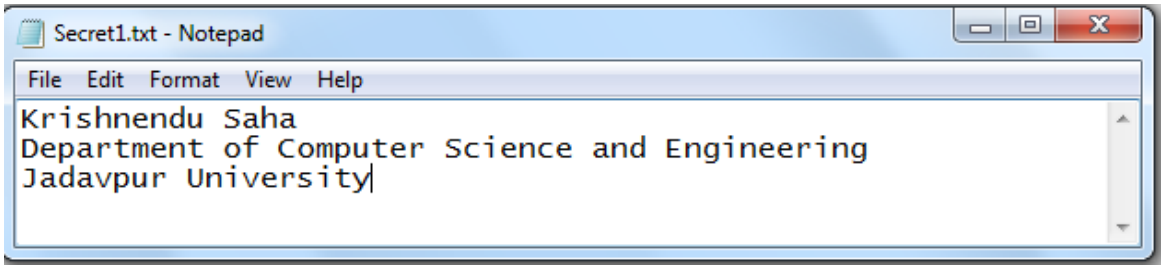
5 Experimental Result Analysis

5.1 Secret Text Multi-Sharing using our Proposed Scheme

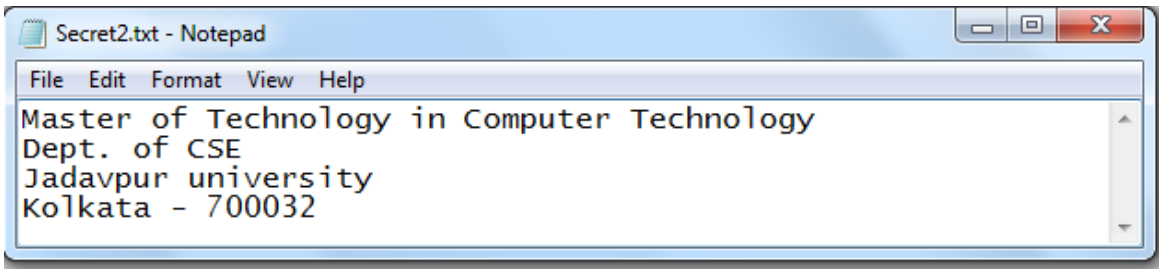
5.1.1 Sharing Phase

In this section, we shall discuss some experimental results for this Multiple Secret Steganography Scheme. Here we share 4 different text shares using this proposed steganography technique.

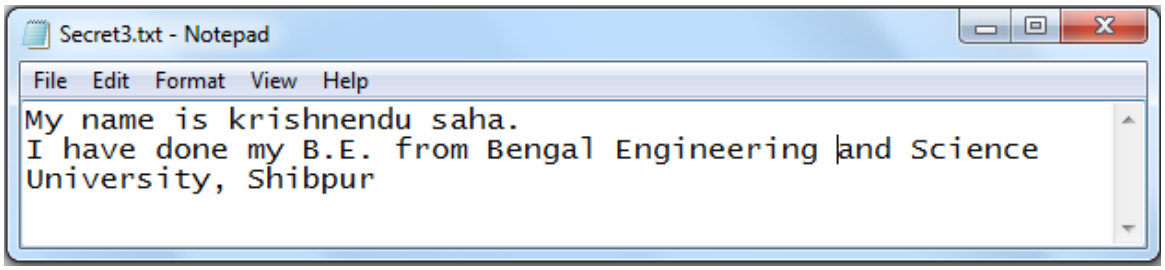
1st Secret Text Message:



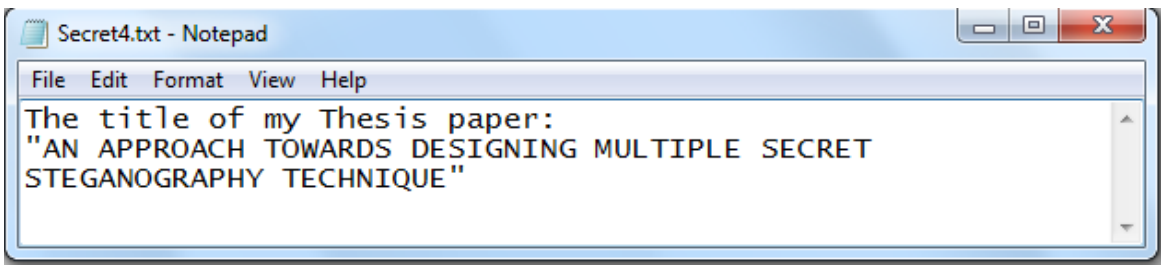
2nd Secret Text Message:



3rd Secret Text Message:



4th Secret Text Message:



These 4 secrets are shared among 4 groups. Following are the details for each group.

Group-1:

Number of members (N1) : 4

Minimum number of shares required to reconstruct (Threshold K1): 2

Group Key (gK1): AAAAAAAAAAAAAAA

Group-2:

Number of members (N2) : 4

Minimum number of shares required to reconstruct (Threshold K2): 3

Group Key (gK2): BBBBBBBBBBBBBBB

Group-3:

Number of members (N3) : 5

Minimum number of shares required to reconstruct (Threshold K3): 3

Group Key (gK3): CCCCCCCCCCCCCCC

Group-4:

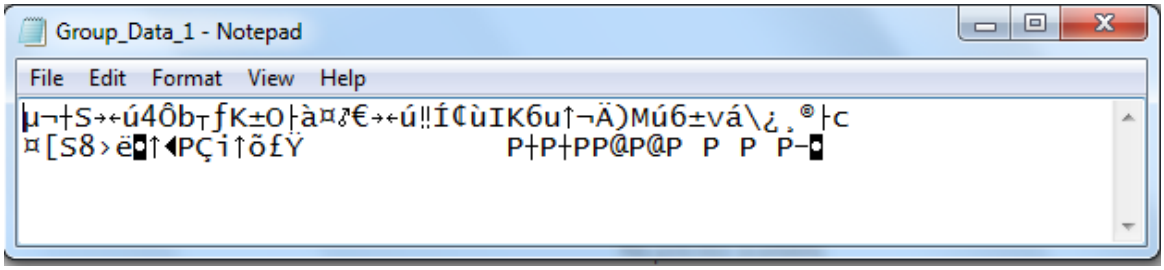
Number of members (N4) : 6

Minimum number of shares required to reconstruct (Threshold K4): 4

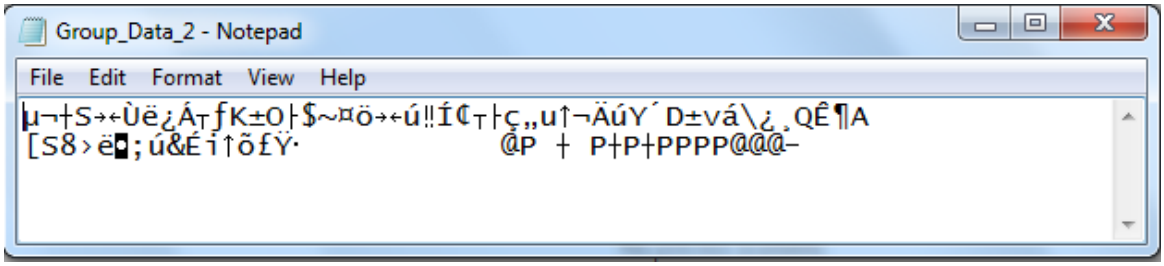
Group Key (gK4): DDDDDDDDDDDDDDD

Here number of intermediate shares N=6. As per our proposed scheme, we know that N is the highest among N1, N2, N3, N4. Intermediate shares for a particular group (say group 2) is shown as follows.

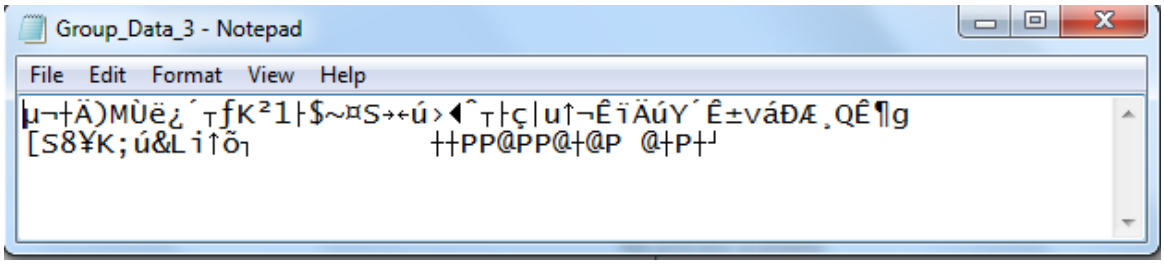
Intermediate share1 Group2 (S21):



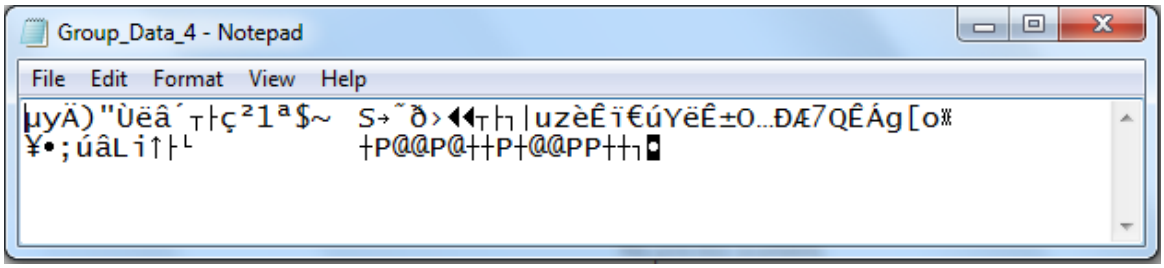
Intermediate share2 Group2 (S22):



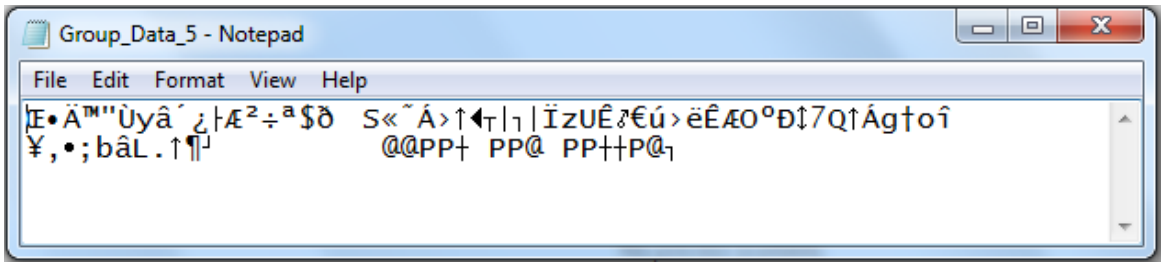
Intermediate share3 Group2 (S23):



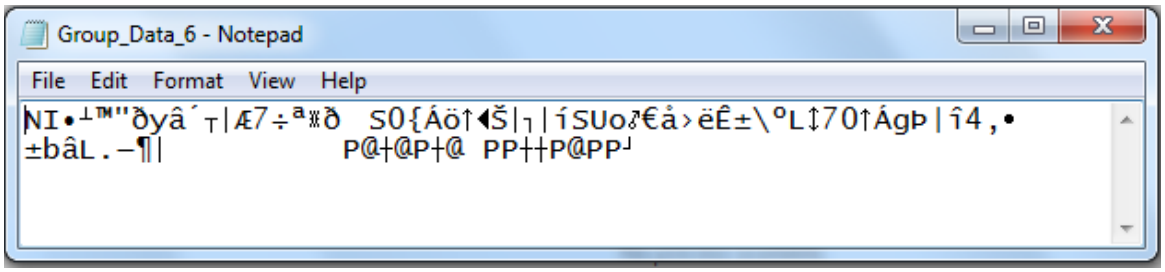
Intermediate share4 Group2 (S24):



Intermediate share5 Group2 (S25):



Intermediate share6 Group2 (S26):



From the above mentioned encrypted intermediate shares, it is very hard to make out any meaningful information related to original message. In this way, all the intermediate shares are generated from all the secrets. i.e.

From Secret Text Message-1: (S11,S12,S13,S14,S15,S16)

From Secret Text Message-2: (S21,S22,S23,S24,S25,S26)

From Secret Text Message-3: (S31,S32,S33,S34,S35,S36)

From Secret Text Message-4: (S41,S42,S43,S44,S45,S46)

As value of N=6, so we need 6 envelope images to insert these intermediate shares. Each envelope image contains one intermediate share for each secrets i.e. first envelope image contains (S11, S21, S31, S41). Similarly all the intermediate shares are stored inside the 6 envelope images in the following way.

Envelope Image-1 contains: (S11, S21, S31, S41)

Envelope Image-2 contains: (S12, S22, S32, S42)

Envelope Image-3 contains: (S13, S23, S33, S43)

Envelope Image-4 contains: (S14, S24, S34, S44)

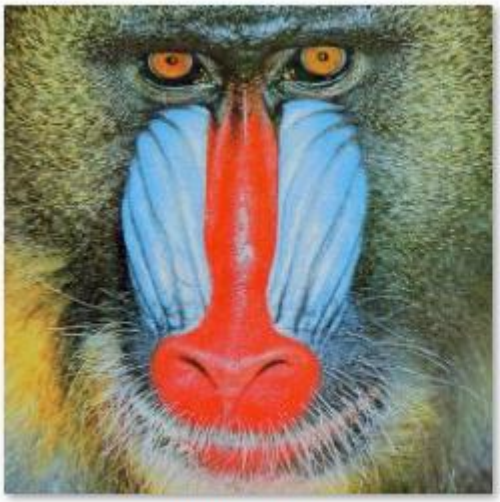
Envelope Image-5 contains: (S15, S25, S35, S45)

Envelope Image-6 contains: (S16, S26, S36, S46)

All the 6 cover images (Before insertion of intermediate shares) are shown as follows.

airplane.bmp

lenna.bmp



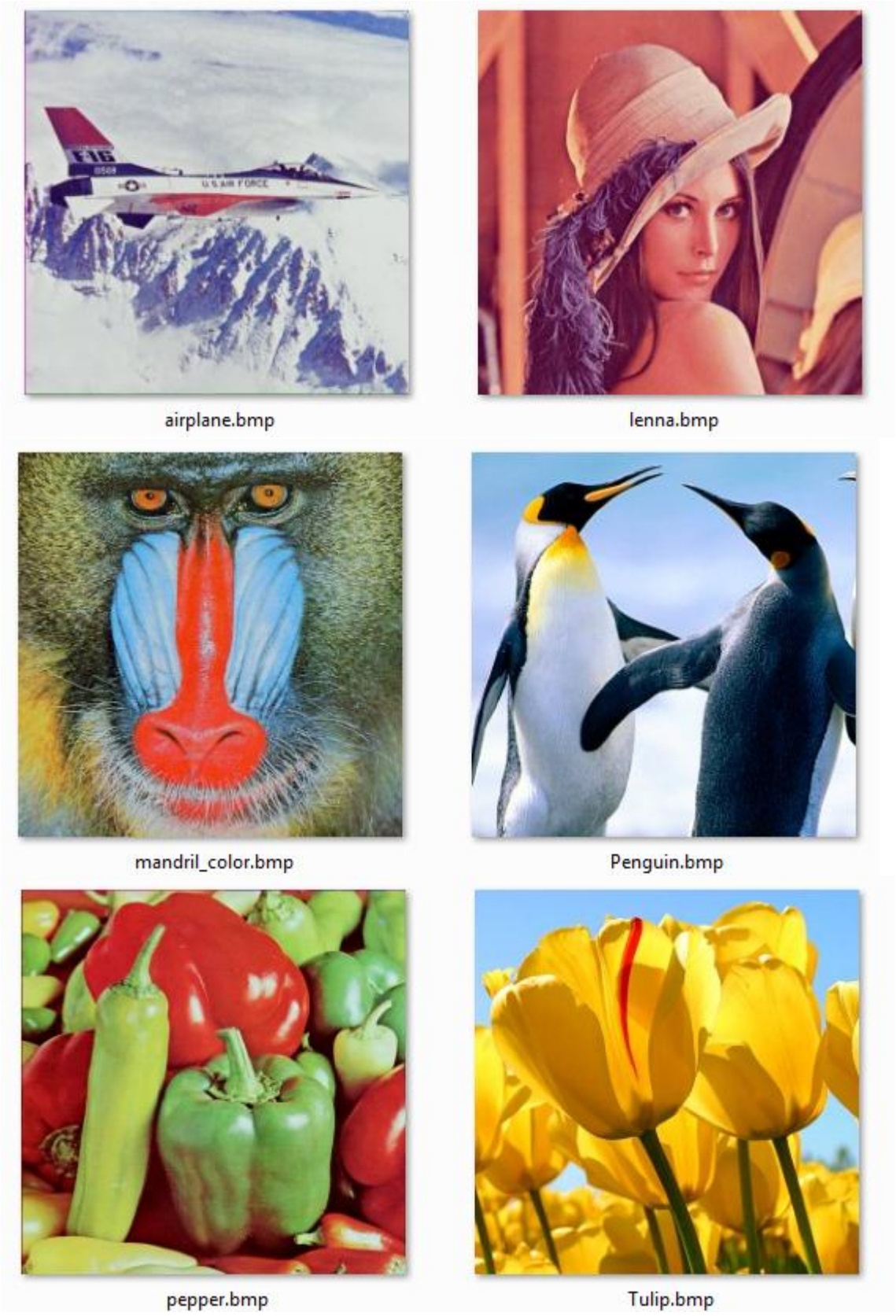
mandril_color.bmp

Penguin.bmp

pepper.bmp

Tulip.bmp

Final Shares (After insertion of intermediate shares) are shown as follows.



Here we found no visible distortion after the steganography operations.

5.1.2 Reconstruction Phase

Reconstruction of secrets from steganography images requires two inputs.

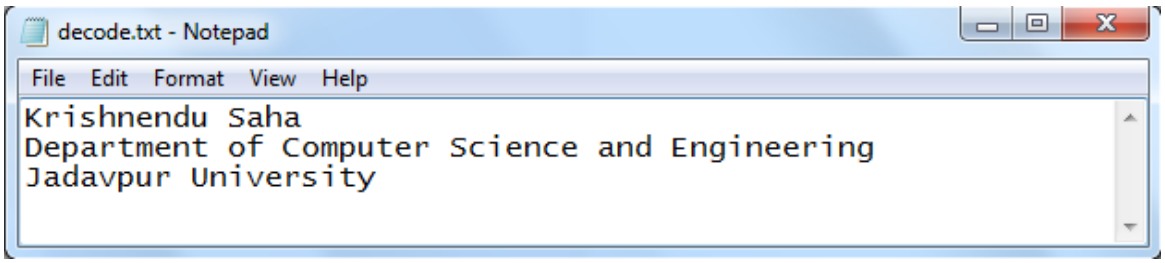
a. A Group Key i.e. unique identifier for a group (Say Group key for group-1 is gK1=AAAAAAAAAAAAAAA)
b. Number of available shares should be more than or equal to the threshold value K of that specific group (Say Threshold value for group-1 is 2)

Reconstruction of the secret for the first group:

Group Key: AAAAAAAAAAAAAAA

Available number of Shares: 2 (i.e. equal to threshold value for group-1)

Decrypted text message for group-1:



Now we shall see what happens when we try to reconstruct the message when number of shares are less than the threshold value.

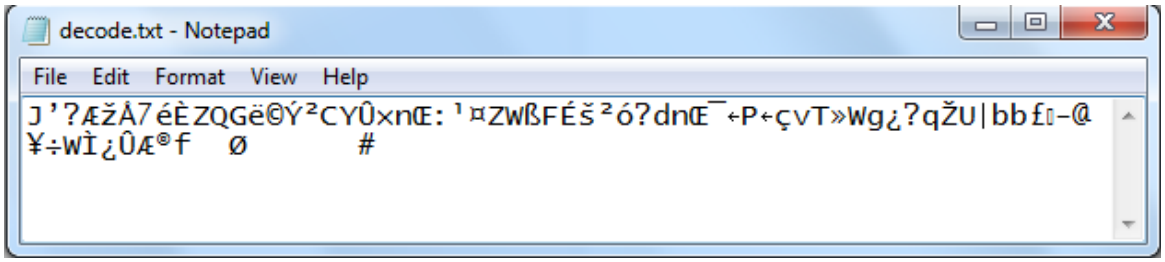
Reconstruction of the secret for the first group:

Group Key: AAAAAAAAAAAAAAA

Available number of Shares: 1 (i.e. Less than the threshold value for group-1)

airplane.bmp

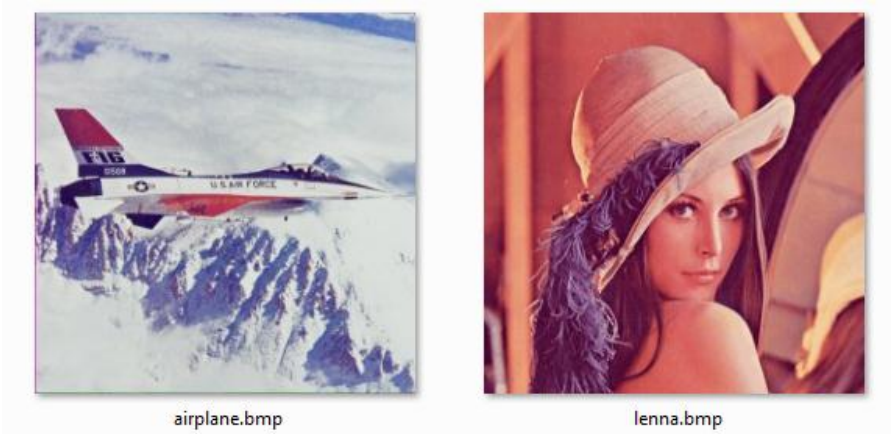
Decrypted text message is as follows



Reconstruction of the secret for the Second group:

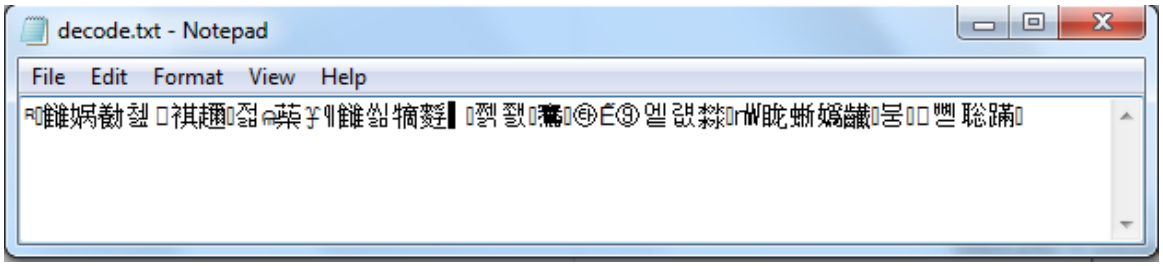
Group Key: BBBBBBBBBBBBBBB

Available number of Shares: 2 (i.e. Less than the threshold value for group-2 as K2=3)



airplane.bmp lenna.bmp

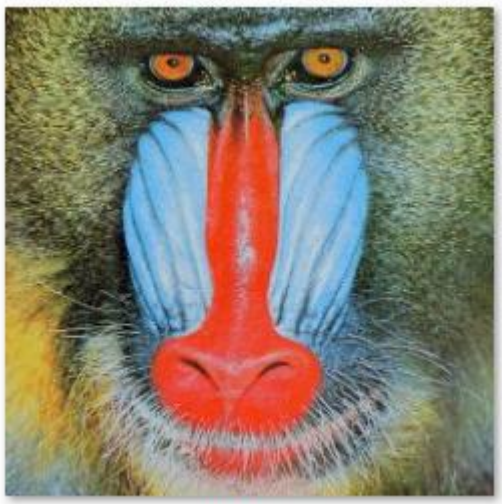
Decrypted text message is shown below



But when we use following 3 shares (Threshold K2=3) for decryption

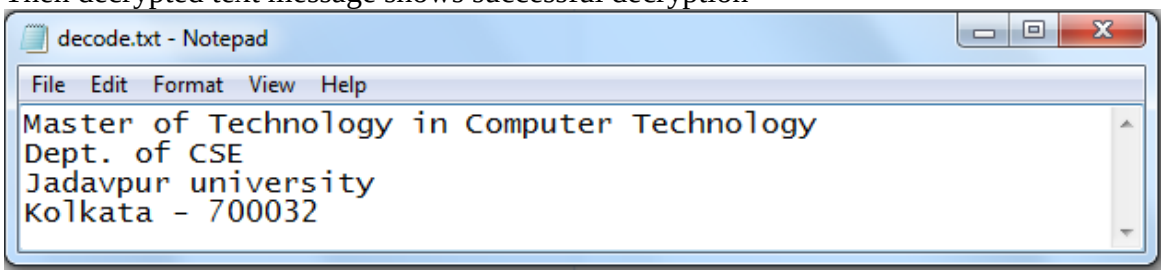
airplane.bmp

lenna.bmp



mandril_color.bmp

Then decrypted text message shows successful decryption



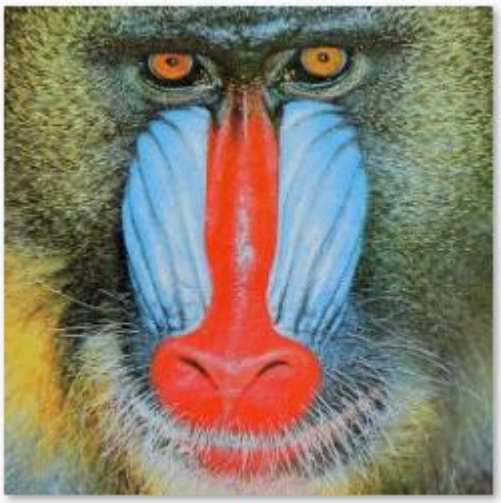
Reconstruction of the secret for the Third group:

Group Key: CCCCCCCCCCCCCCC

Available number of Shares: 3 (i.e. Equal to the threshold value for group-3 as K3=3)

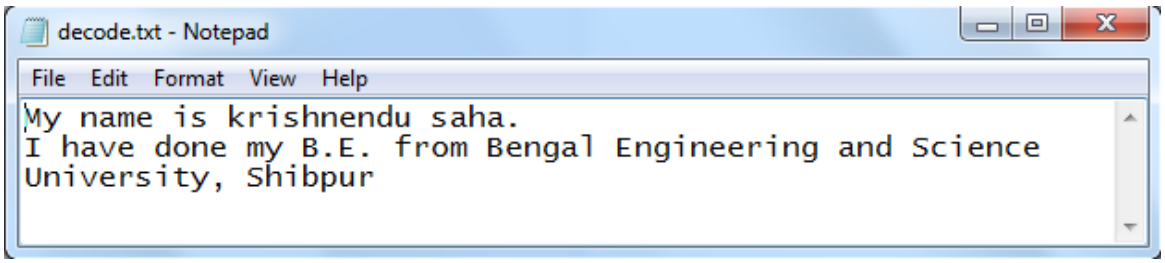
airplane.bmp

lenna.bmp



mandril_color.bmp

Decrypted text message is as follows



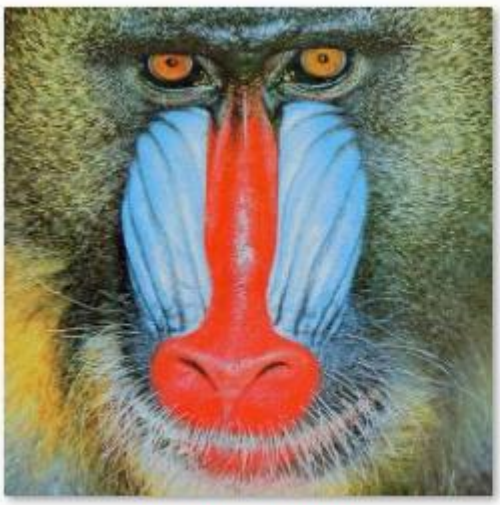
Reconstruction of the secret for the Fourth group:

Group Key: DDDDDDDDDDDDDDDD

Available number of Shares: 4 (i.e. Equal to the threshold value for group-4 as K4=4)

airplane.bmp

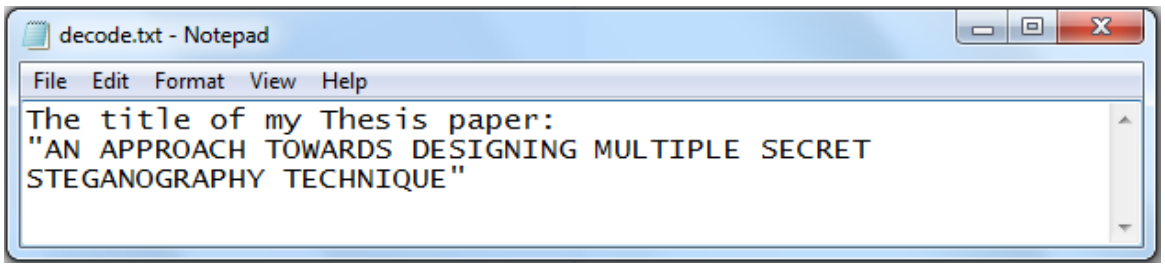
lenna.bmp



mandril_color.bmp

Penguin.bmp

Decrypted text message is as follows



5.2 Secret Image Multi-Sharing using our Proposed Scheme

Following are the three secret images that we want to share using this Multiple secret Steganography Scheme.

Butterfly (96x96).jpg

Cactus (120x100).jpg

Sunflower (120x120).jpg

5.2.1 Sharing Phase

These 3 images are shared among three groups. Group information is as follows.

Group-1:

Number of members (N1) : 4

Minimum number of shares required to reconstruct (Threshold K1): 3

Group Key (gK1): XXXXXXXXXXXXXXX

Group-2:

Number of members (N2) : 5

Minimum number of shares required to reconstruct (Threshold K2): 3

Group Key (gK2): YYYYYYYYYYYYYYY

Group-3:

Number of members (N3) : 6

Minimum number of shares required to reconstruct (Threshold K3): 4

Group Key (gK3): ZZZZZZZZZZZZZZZ

Intermediate shares are generated from each group is as follows
Group-1: (S11, S12, S13, S14, S15, S16)
Group-2: (S21, S22, S23, S24, S25, S26)
Group-3: (S31, S32, S33, S34, S35, S36)

We need total 6 cover images as N={highest among N1, N2, N3}=6. Intermediate shares are distributed in these cover images in the following way.
Cover Image-1: (S11, S21, S31)
Cover Image-2: (S12, S22, S32)
Cover Image-3: (S13, S23, S33)
Cover Image-4: (S14, S24, S34)
Cover Image-5: (S15, S25, S35)
Cover Image-6: (S16, S26, S36)

Following are the **cover images (512x512 pixels) before insertion** of intermediate shares

BoatsColor.bmp

lenna.bmp

mandril_color.bmp

Penguin.bmp

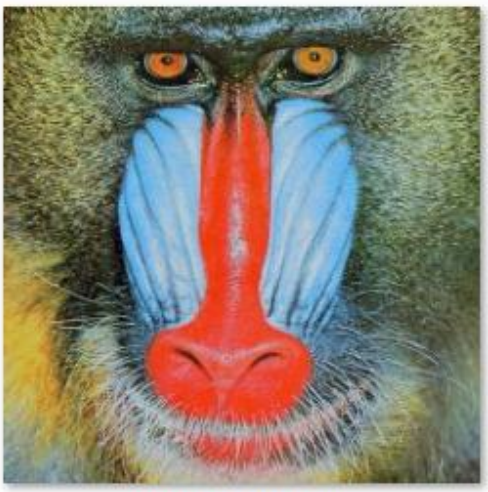
pepper.bmp

Tulip.bmp

Following are the **Final Shares (512x512 pixels) after insertion** of intermediate shares

BoatsColor.bmp

lenna.bmp



mandril_color.bmp

Penguin.bmp

pepper.bmp

Tulip.bmp

5.2.2 Reconstruction phase

Reconstruction for Group-1:
Group Key: XXXXXXXXXXXXXXX
Threshold value: 3
Shares used for reconstruction:

a. BoatsColor.bmp
b. lenna.bmp
c. mandril_color.bmp

Reconstructed Image:

Reconstruction for Group-2:
Group Key: YYYYYYYYYYYYYYY
Threshold value: 3
Shares used for reconstruction:

a. BoatsColor.bmp
b. lenna.bmp
c. mandril_color.bmp

Reconstructed Image:

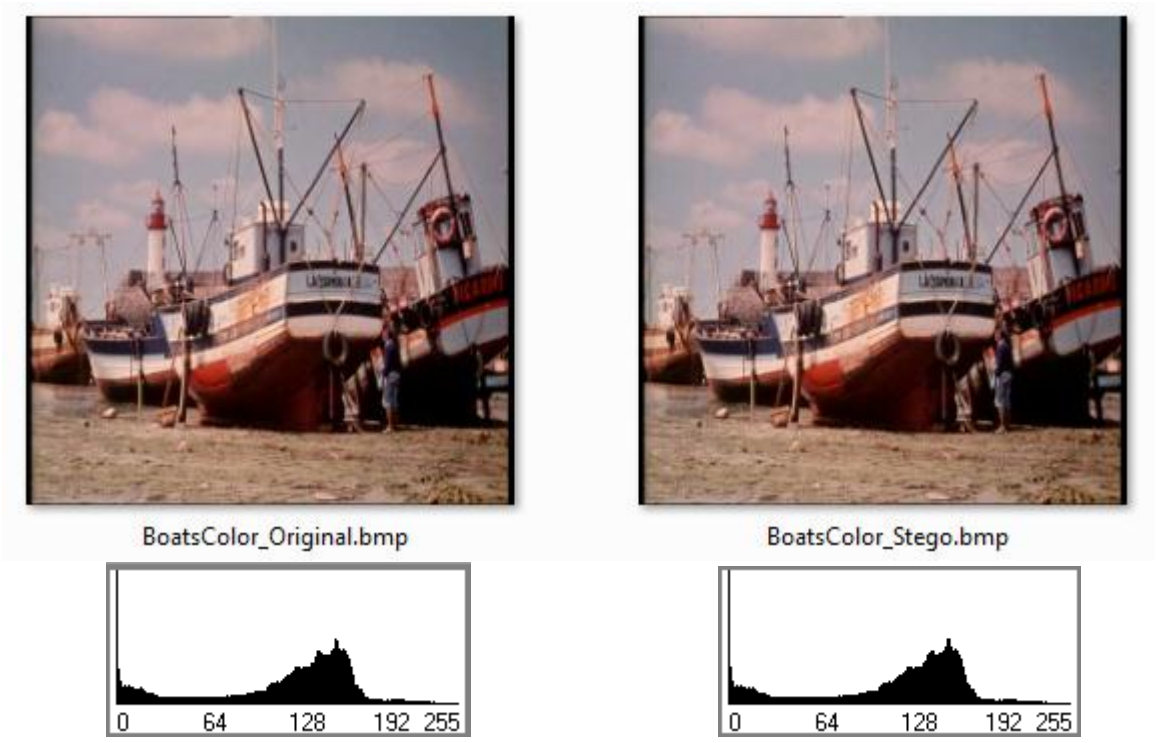
Reconstruction for Group-3:
Group Key: ZZZZZZZZZZZZZZZ
Threshold value: 4
Shares used for reconstruction:

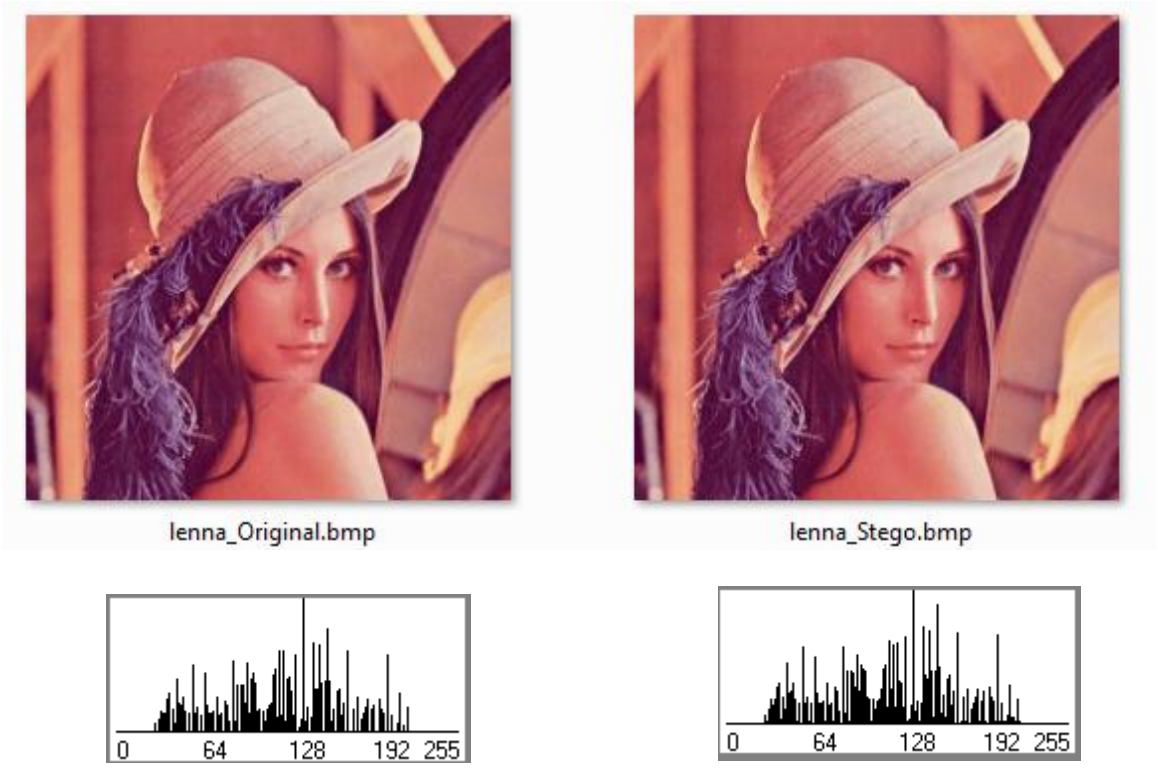
a. BoatsColor.bmp
b. lenna.bmp
c. mandril_color.bmp
d. Penguin.bmp

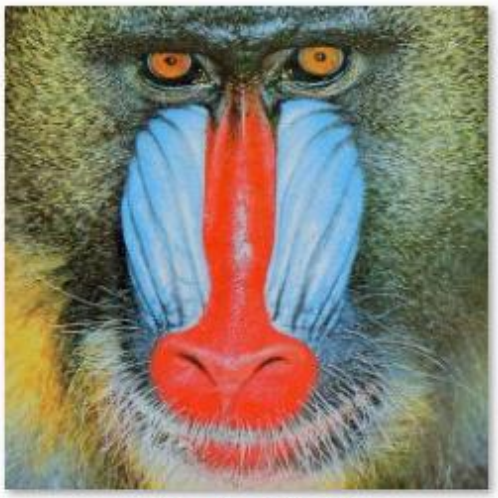
Reconstructed Image:

5.3 Histogram data analysis

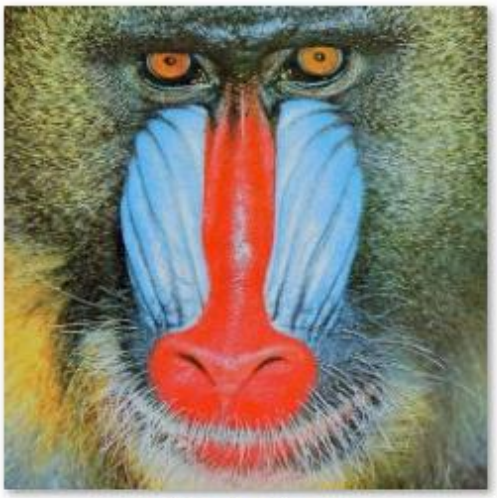
Here we shall discuss Image quality comparison of Original cover image and Steganography images based on histogram analysis for the Image Multi-Secret Sharing experiment mentioned in the previous section.



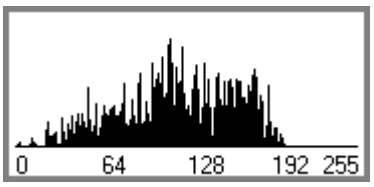


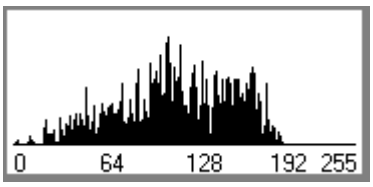


mandril_color_Original.bmp



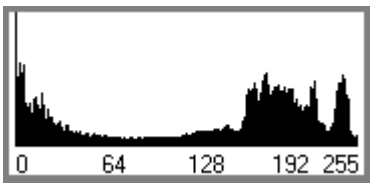
mandril_color_Stego.bmp

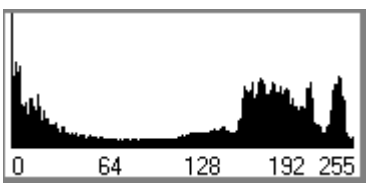




Penguin_Original.bmp

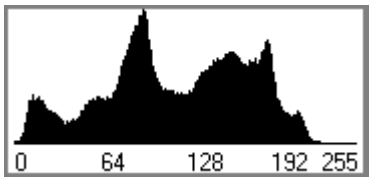
Penguin_Stego.bmp

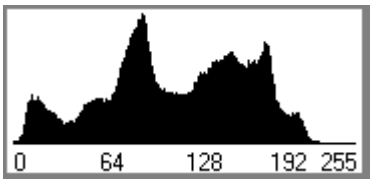




pepper_Original.bmp

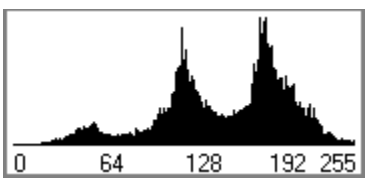
pepper_Stego.bmp

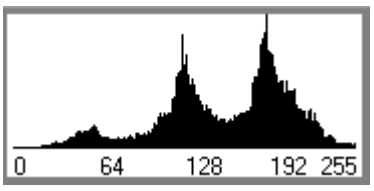




Tulip_Original.bmp

Tulip_Stego.bmp





From the above mentioned histogram analysis, we can conclude that there is no visible change in the histogram data due to insertion of secret messages into it.

5.4 Key Sensitivity Analysis

We have already seen in the experimental result in section 5.2 that we are able to reconstruct the secret successfully if we have proper group key and number of available shares greater than or equal to the threshold value of that group. We have already seen following result while reconstructing the secret for Group-1 data.

Reconstruction for Group-1:
Group Key: XXXXXXXXXXXXXXX
Threshold value: 3
Shares used for reconstruction:

a. BoatsColor.bmp
b. lenna.bmp
c. mandril_color.bmp

Reconstructed Image:

Now we alter a single byte in the group key keeping rest of the conditions unchanged, i.e.

Reconstruction for Group-1:
Group Key: XX**C**XXXXXXXXXXXXX
Threshold value: 3
Shares used for reconstruction:

a. BoatsColor.bmp
b. lenna.bmp
c. mandril_color.bmp

With this single byte key change, we found that decrypted data "decode.jpg" have size 0. i.e. nothing got decrypted while single byte key got changed. Similar result observed with group-3 secret decryption.

Reconstruction for Group-3:
Group Key: ZZZZZZZZZZ**8**ZZZZZ
Threshold value: 4
Shares used for reconstruction:

a. BoatsColor.bmp
b. lenna.bmp
c. mandril_color.bmp
d. Penguin.bmp

Reconstructed Image: "decode.jpg" is of size 0 KB.

5.5 Histogram and PSNR Analysis with Max. Sharing Load

Maximum 9 secrets can be shared using our proposed Multiple secret steganography scheme. This limitation is to ensure there is no distortion of cover image due to steganography operations. Following are the 9 secret images that needs to be shared.

Secret Image dimensions:

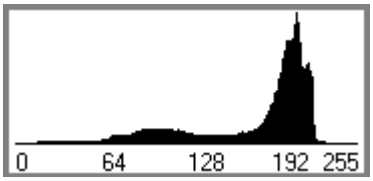
- Apple.jpg – 100 x 100 pixels
- Boat.jpg – 100 x 80 pixels
- Butterfly.jpg – 96 x 96 pixels
- Cactus.jpg – 120 x 100 pixels
- CoconutTree.jpg – 100 x 100 pixels
- Flower.jpg – 120 x 110 pixels
- LordDurga.jpg – 80 x 120 pixels
- Rose.jpg – 110 x 80 pixels
- Sun_Flower.jpg – 120 x 120 pixels

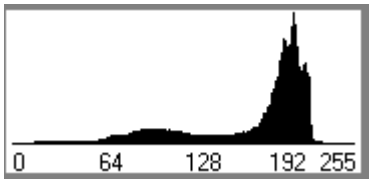
Group Information:
Group-1: N=4, K=3, Group Key = AAAAAAAAAAAAAAAA Secret= Apple.jpg
Group-2: N=4, K=3, Group Key = BBBBBBBBBBBBBBB Secret= Boat.jpg
Group-3: N=4, K=3, Group Key = CCCCCCCCCCCCCCC Secret= Butterfly.jpg
Group-4: N=4, K=3, Group Key = DDDDDDDDDDDDDDD Secret= Cactus.jpg
Group-5: N=4, K=3, Group Key = EEEEEEEEEEEEEEE Secret= CoconutTree.jpg
Group-6: N=4, K=3, Group Key = FFFFFFFFFFFFFFF Secret= Flower.jpg
Group-7: N=4, K=3, Group Key = GGGGGGGGGGGGGGG Secret= LordDurga.jpg
Group-8: N=4, K=3, Group Key = HHHHHHHHHHHHHHH Secret= Rose.jpg
Group-9: N=4, K=3, Group Key = IIIIIIIIIIIIII Secret= Sun_Flower.jpg

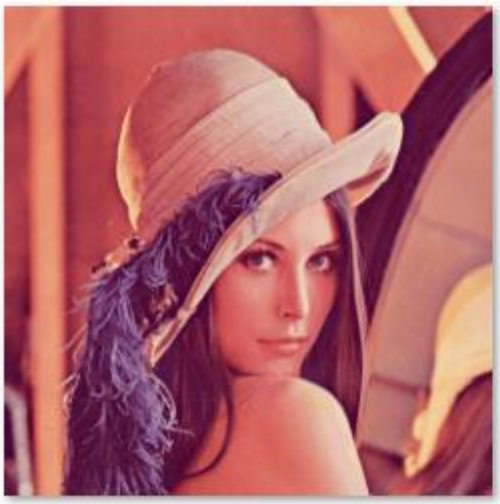
Here we shall discuss about image quality distortion due to insertion of secret messages into the cover images through Histogram and PSNR analysis.

airplane_Original.bmp

airplane_stego.bmp

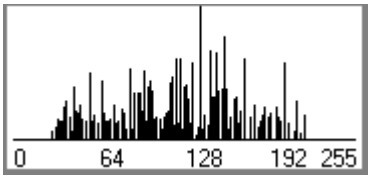


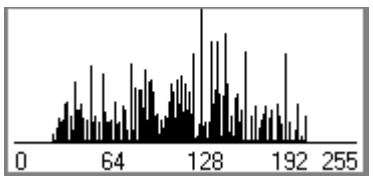


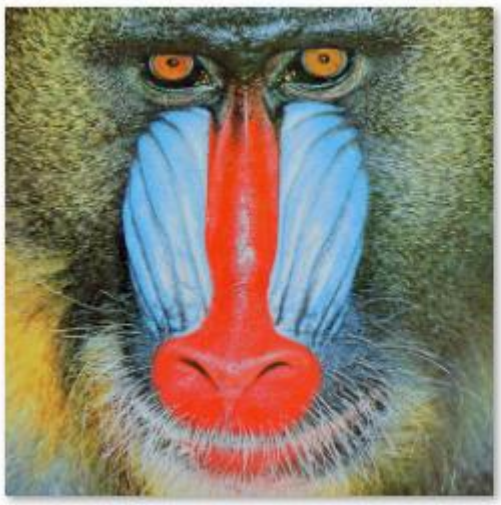


lenna_Original.bmp

lenna_Stego.bmp

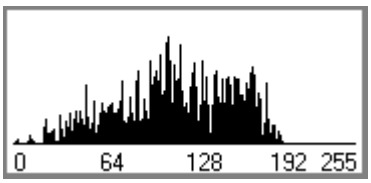


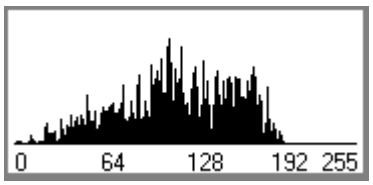




mandril_color_Original.bmp

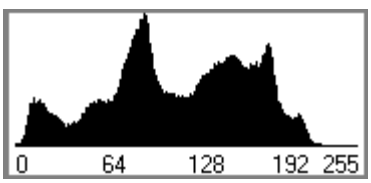
mandril_color_Stego.bmp

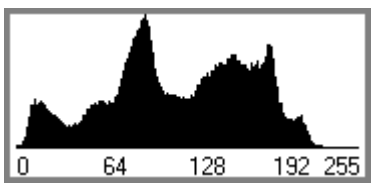




pepper_Original.bmp

pepper_Stego.bmp





PSNR Analysis with Max. Sharing load:

Maximum sharing load implies secret data is stored in all the three color channels (Red, Green, Blue) and three LSB positions for each channel.

Original Image	**Steganography Image**	**PSNR for Red Channel**	**PSNR for Green Channel**	**PSNR for Blue Channel**
airplane_Original.bmp	airplane_stego.bmp	46.9868	47.6495	46.8515
lenna_Original.bmp	lenna_Stego.bmp	47.3093	47.8923	46.9819
mandril_color_Original.bmp	mandril_color_Stego.bmp	47.5558	48.7050	46.9608
pepper_Original.bmp	pepper_Stego.bmp	46.9748	47.6307	46.7724

PSNR Analysis for the image secret sharing example in section 5.2:

Here only one LSB position is used per channel (Red, Green, Blue) to store the secret message.

Original Image	**Steganography Image**	**PSNR for Red Channel**	**PSNR for Green Channel**	**PSNR for Blue Channel**
BoatsColor_Original.bmp	BoatsColor_Stego.bmp	59.3989	58.0757	60.2052
lenna_Original.bmp	lenna_Stego.bmp	59.5868	58.0444	60.1125
mandril_color_Original.bmp	mandril_color_Stego.bmp	59.3841	58.1111	60.2641
Penguin_Original.bmp	Penguin_Stego.bmp	59.4025	58.1318	60.2162
pepper_Original.bmp	pepper_Stego.bmp	59.3898	58.1014	60.2530
Tulip_Original.bmp	Tulip_Stego.bmp	59.3823	58.1093	60.1616

6 Strength, Applications & Future Extensions

6.1 Strength of Our Protocol

This proposed Multiple Secret Sharing scheme is one of the simplest threshold sharing scheme, practically having minimal computational overhead during both share generation and reconstruction phases. In various secret sharing schemes, if number of shares is close to threshold value, then reconstructed file may be quite closer to the original secret file. But in our proposed scheme if and only if numbers of collating shares are equal to k (threshold) or more, then only the original secret message is reconstructed; otherwise reconstructed message will be completely cipher from the original secret message. Because fewer shares cannot reconstruct the original header, thus we cannot have either right key or the information to construct the correct masking pattern. So our proposed scheme is completely secured and also confirms confidentiality and integrity as well.

Not only that, this scheme is equally applicable for any file format such as Image, Text and Audio etc. and only when legitimate group of qualified set of shares are come together for a particular group, then only we can reconstruct the original secret message.

Also reconstruction of secret for a specific group require two inputs, a Group Key and number of shares at least equal to the number of threshold value of that specific group. We have already shown in key sensitivity analysis that if we change a single byte in group key, then also it is not possible to reconstruct anything using that key.

6.2 Applications

Research in steganography has mainly been driven by a lack of strength in cryptographic systems. Many governments have created laws to either limit the strength of a cryptographic system or to prohibit it altogether, forcing people to study other methods of secure information transfer. Businesses have also started to realize the potential of steganography in communicating trade secrets or new product information. Avoiding communication through well-known channels greatly reduces the risk of information being leaked in transit. Hiding information in a photograph like company picnic photo is less suspicious than communicating an encrypted file.

Thus this proposed secured Steganography scheme can be used for a wide range of applications such as, in defense organizations for safe circulation of secret data, in military and intelligence agencies, in smart identity cards where personal details are embedded in the photograph itself for copyright control of materials. In medical imaging, patient's details are embedded within image providing protection of information and reducing transmission time and cost, in online voting system so as to make the online election secure and robust against a variety of fraudulent behaviors, for data hiding in countries where cryptography is prohibited, in improving mobile banking security, in tamper proofing so as to prevent or detect unauthorized modifications and numerous other applications.
Steganography can also be misused like other technologies. For instance terrorists may use this technique for their secret secure communication or anti-virus systems can be fooled if viruses are transmitted in this way. However, it is evident that steganography has numerous useful applications and will remain the point of attraction for researchers.

6.3 Future Extension

This paper presented the research work in the field of Image steganography solely focusing in spatial domain. In future we can extend this steganography scheme in transform and compression domains for the digital images. Transform domain techniques make changes in the frequency coefficients instead of manipulating the image pixels directly, thus distortion is kept at minimum level and that's why they are preferred over spatial domain techniques. But when it comes to embedding capacity, spatial domain techniques give better results. However, there exists a trade-off between the image quality and the embedding capacity. Hiding more data in spatial domain results directly into more distortion of the image. So the deployment of steganography technique is dependent on the type of application it is designed for. Also in recent times, some researchers have concentrated on embedding secret data into the compression codes of images. Such need arises keeping in mind the bandwidth requirements. We have plan to extend this Multi-Secret steganography also in compression domain.

7 Conclusion

Steganography transmits secrets through apparently innocuous covers in an effort to conceal the existence of a secret. Digital image steganography and its derivatives are growing in use and applications in the areas where cryptography and strong encryption are being outlawed, people are looking at steganography to circumvent such policies and pass messages covertly. As with the other great innovations in the digital age, the battle between cryptographers and cryptanalysis, security experts and hackers, record companies and pirates, steganography and Steganalysis will continually develop new techniques to counter each other. Being a part of this continuous development process, in our current endeavor we tried to extend our proposed Visual Multiple Secret Sharing scheme to multiple secret steganography scheme with some modification to make the scheme more robust and secure. In this proposed scheme, multiple secret messages can be shared across multiple groups using steganography techniques and successful reconstruction possible if and only if specific group key and number of shares more than or equal to number of threshold value of that group is available. On the other hand, no secret data can be reconstructed if either used wrong group key or number of shares less than the threshold value of that group. Also all generated shares contain partial ciphered information from each of the secret message. That provides strong protection of the secret messages. Even this proposed mechanism has highest level key sensitivity such that any single byte change in key value will result in failure to reconstruct any secret information. So our proposed scheme is a Perfect Secret Sharing Scheme (PSS) which applies Spatial Image Steganography in an efficient way.

References

[1] A. Shamir: “How to share a secret ?” Comm ACM, 22(11):612-613, 1979.

[2] G. Blakley : “Safeguarding cryptographic keys “ Proc. of AFIPS National Computer Conference, 1979.

[3] C. Asmuth and J. Bloom :”A modular approach to key safeguarding” IEEE transaction on Information Theory, 29(2):208-210, 1983.

[4] E. D. Karnin, J. W. Greene, and M. E. Hellman, “On secret sharing systems”, vol. IT-29, no. 1, pp. 35-41, Jan 1983.

[5] Hide & Seek: An Introduction to Steganography: Niels Provos and Peter Honeyman, IEEE Security & Privacy Magazine, May/June 2003.
http://niels.xtdnet.nl/papers/practical.pdf

[6] Steganography and steganalysis-Robert Krenn, Internet Publication, March 2004 http://www.krenn.nl/univ/cry/steg/article.pdf

[7] Steganographic Techniques and their use in an Open-Systems Environment- Bret Dunbar, The Information Security Reading Room, SANS Institute 2002 http://www.sans.org/reading-room/whitepapers/covert/677.php

[8] Steganography Primer - Ruid, Computer Academic underground,2004 http://www.dustintrammell.com/presentations/Steganography-Primer.pdf

[9] Moerland, T., “Steganography and Steganalysis”, Leiden Institute of Advanced Computing Science, www.liacs.nl/home/tmoerl/privtech.pdf

[10] Silman, J., “Steganography and Steganalysis: An Overview”, SANS Institute, 2001

[11] "Steganography An Art of Hiding Data", Shashikala Channalli et al /International Journal on Computer Science and Engineering Vol.1(3), 2009, 137-141

[12] Bozkurt, Kaya, Selcuk, Guloglu “Threshold Cryptography Based on Blakely Secret Sharing” Information Sciences.

[13] C.-C. Thien and J.-C. Lin, “Secret image sharing”, Computers and Graghics, vol. 26, no. 5, pp. 765-770, 2002.

[14] K. Kaya and A. A. Selcuk "Threshold Cryptograhy based on Asmuth-Bloom Secret Sharing" Information Sciences 177(19) 4148-4160, 2007

[15] M. Naor and A. Shamir, Visual Cryptography, Advance in Cryptography, Eurocrypt'94, Lecture Notes in Computer Science 950, pp. 1-12, Springer Verlag, 1994.

[16] A. Adhikari and B. Roy, On some Constructions of Monochrome Visual Cryptographic Schemes, 1st International Conference on Information Technology, IT 2008, Poland, May 2008.

[17] A. Adhikari, M. Bose, D. Kumar and B. Roy, Applications of Partially Balanced and Balanced Incomplete Block Designs in developing Visual Cryptographic Schemes, IEICE TRANS. FUNDAMENTALS, Japan, Vol. E-90A, No. 5, pp. 949-951, May 2007.

[18] Mizuho Nakajima and Yasushi Yamaguchi, Extended Visual Cryptography for Natural Images.

[19] G. Ateniese, C. Blundo, A. De Santis and D. Stinson, Visual cryptography for general access structures, Information and Computation, 129(2):86-106, 1996.

[20] M. Naor and A. Shamir, Visual Cryptography ii : Improving the contrast via the cover base, Theory of Cryptography Library, (96-07), 1996.

[21] MD5 in wikipedia link "http://en.wikipedia.org/wiki/MD5".

[22] "Image Secret Sharing Scheme using a Novel Secret Sharing Technique with Steganography"-IEEE CASCOM Post Graduate Student Paper Conference, Nov.27.2010, Page- 62-65.
http://www.ewh.ieee.org/r10/calcutta/cas/Report_IEEE_CASCOM_PGSPC_2010.pdf

[23] "A Novel Image Secret Sharing Scheme" –Emerging Application of Information Technology (EAIT), 2011 Second International Conference, Kolkata. DOI:10.1109/EAIT.2011.43, Page-177-180.
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5734922

[24] "Ultra Secured and Authentic Key Distribution Protocol using a Novel Secret Sharing Technique"- International Journal of Computer Applications (0975-8887) (IJCA), Volume 19-No.7, April 2011, DOI: 10.5120/2375-3179, Page-12-15.
http://www.ijcaonline.org/volume19/number7/pxc3873129.pdf

Appendix

Appendix A

A.1 Message Digest Version (MD 5)

In this project we have used MD5[21] algorithm to created 128-bits key from user defined variable length key.

Message-Digest algorithm 5 is a widely used cryptographic hash function with a 128-bit hash value. Specified in RFC 1321, MD5 has been employed in a wide variety of security applications and is also commonly used to check the integrity of files.

It is one in a series of message digest algorithm designed by Professor Ronald Rivest of MIT (Rivest, 1994). When analytic work indicated that MD5's predecessor MD4 was likely to be insecure, MD5 was designed in 1991 to be a secure replacement. (Weaknesses were indeed later found in MD4 by Hans Dobbertin.)

The algorithm takes as input a message of arbitrary length and produces as output a 128 bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given pre-specified target message digest. The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly.

MD5 Hashes

The 128-bit (16-byte) MD5 hashes (also termed message digest) are typically represented as a sequence of 32 hexadecimal digits. The following demonstrates a 43-byte ASCII input and the corresponding MD5 hash:

MD5 ("Jadavpur University")

= 9784924df7c536cbf21e62d223958e33

Even a small change in the message will (with overwhelming probability) result in a completely different hash, due to the avalanche effect. For example, adding a period to the end of the sentence:

MD5 ("Jadavpur University**.**")

= b22e4231ed88993b67413b66d7f52a87

The hash of the zero-length string is: MD5 ("")

= d41d8cd98f00b204e9800998ecf8427e

Appendix B

B.1 Hardware and Software Requirements of the project

(i) Microsoft Visual Studio 2010 IDE

(ii) Operating System: Microsoft Windows 7

(iii) Microsoft .NET Framework Version 4.0 (for Graphical User Interface).

(iv) Intel Core i3 Processor with 4GB Main Memory.

B.2 Minimum System Requirements

(i) Any Window Operating System with 32bit Processor.

(ii) Microsoft Dot NET 4.0 or above version

(ii) 100 MB system memory.

Appendix C

C.1 Format of Bitmap Image (.BMP File)

A digital image is represented by a one dimensional array of numbers that represent the different light intensities of each pixel. The dimension of a 640 × 480 pixel image can literally be multiplied out to find the total number of pixel in the image, in this case 307,200 pixels. In digital photography this is known as the resolution of an image**,** a digital camera that takes pictures of 640 × 480 is known as a 0.3 Mega Pixel resolution camera for this reason.

Digital images usually use either 24-bits (standard bitmap) or 8 bits (standard GIF image, color or grey scale) for the storage of intensity information per pixel. This means that in a bitmap image (BMP) there are a potential 16.8 Million colors (224) per pixel and in a GIF, 256 different color combinations.

In a typical 24-bit image, each pixel has three color components, red, green and blue, each component using 8 bits to represent a value from 0 to 255. An 8 bit image on the other hand can either have a color palette of 256 different grey levels or color values. Naturally this results in the 8 bit color image having to perform a "best fit" in order to match a real world color to its limited palate.

The number of pixels in an uncompressed BMP image contributes directly to file size, for instances, a 640 × 480 image has 3,07,200 pixels in total, and each of this is represented by 24 bits which equals a total of 900 Kilobytes.

Due to this size overhead images are often compressed using either lossy or lossless compression. Lossy compression as the name suggests, reduces the file size but at the expense of the image's integrity, JPEG (Joint Photographic Experts Group) format is a prime example of this, trading quality of image for level of compression.

Lossless compression is sometimes used to save space while maintaining the images integrity as it always reconstructs the image exactly as it was before compression.

In bitmap images this is usually Run Length Encoding, succinctly describes it as method consisting of the process of searching for repeated runs of a single symbol in an input stream, and replacing them by a single instance of the symbol and a run count. In the case of a bitmap image the symbol it hopes to find runs of are the concurrent 1s or 0s that represent color values.

The BMP file format, sometimes called bitmap or DIB file format (for device-independent bitmap), is an image file format used to store bitmap digital images, especially on Microsoft Windows and OS/2 operating systems.

C.1.1 Pixel Storage

In uncompressed BMP files, and many other bitmap file formats, image pixels are stored with a color depth of 1, 4, 8, 16, 24, or 32 bits per pixel. Images of 8 bits and fewer can be either grayscale or indexed color. An alpha channel (for transparency) may be stored in a separate file, where it is similar to a grayscale image, or in a fourth channels that converts 24-bit images to 32 bits per pixel.

Uncompressed bitmap files (such as BMP) are typically much larger than compressed (with any of various methods) image file formats for the same image. The bits representing the bitmap pixels are packed within rows. Depending on the color depth, a pixel in the picture will occupy at least *n*/8 bytes (*n* is the bit depth, since 1 byte equals 8 bits). The approximate size for a n-bit (2^n colors) BMP file in bytes can be calculated, including the effect of starting each word on a 32-bit word boundary, as:

$$\text{rowsize} = 4 \cdot \left\lfloor \frac{(n \cdot \text{width}) + 31}{32} \right\rfloor,$$

Where the floor function gives the highest integer that is less than or equal to the argument; that is, the number of 32-bit words needed to hold a row of *n*-bit pixels; this value multiplied by 4 gives the byte count.

$$\text{filesize} \approx 54 + 4 \times 2^n + \text{rowsize} \times \text{height},$$

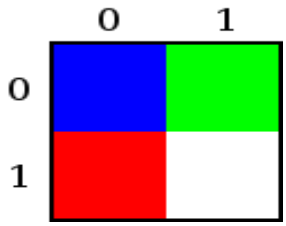
where height and width are given in pixels.

In the formula above, 54 is the size of the headers in the popular Windows V3 BMP version (14-byte BMP file header plus 40-byte DIB V3 header); some other header versions will be larger or smaller than that, as described in tables below. And $4 \cdot 2^n$ is the size of the color palette; this size is an approximation, as the color palette size will be $3 \cdot 2^n$ bytes in the OS/2 V1 version, and some other versions may optionally define only the number of colors needed by the image, potentially fewer than 2^n. Only files with 8 or fewer bits per pixel use a palette; for 16-bit (or higher) bitmaps, omit the palette part from the size calculation:

$$\text{filesize} = 54 + \text{rowsize} \times \text{height}$$

C.1.2 Header Structure

Here we describe 54-bytes header structure with an example of a 2x2 Pixel, 24-Bit Bitmap.



Offset	Size	Hex Value	Value	Description
0h	2	42 4D	"BM"	Magic Number (unsigned integer 66, 77)
2h	4	46 00 00 00	70 Bytes	Size of the BMP file
6h	2	00 00	Unused	Application Specific
8h	2	00 00	Unused	Application Specific
Ah	4	36 00 00 00	54 bytes	The offset where the bitmap data (pixels) can be found.
Eh	4	28 00 00 00	40 bytes	The number of bytes in the header (from this point).
12h	4	02 00 00 00	2 pixels	The width of the bitmap in pixels
16h	4	02 00 00 00	2 pixels	The height of the bitmap in pixels
1Ah	2	01 00	1 plane	Number of color planes being used.
1Ch	2	18 00	24 bits	The number of bits/pixel.
1Eh	4	00 00 00 00	0	BI_RGB, No compression used
22h	4	10 00 00 00	16 bytes	The size of the raw BMP data (after this header)
26h	4	13 0B 00 00	2,835 pixels/meter	The horizontal resolution of the image
2Ah	4	13 0B 00 00	2,835 pixels/meter	The vertical resolution of the image
2Eh	4	00 00 00 00	0 colors	Number of colors in the palette
32h	4	00 00 00 00	0 important colors	Means all colors are important
Start of Bitmap Data				
36h	3	00 00 FF	0 0 255	Red, Pixel (0,1)
39h	3	FF FF FF	255 255 255	White, Pixel (1,1)

3Ch	2	00 00	0 0	Padding for 4 byte alignment (Could be a value other than zero)
3Eh	3	FF 00 00	255 0 0	Blue, Pixel (0,0)
41h	3	00 FF 00	0 255 0	Green, Pixel (1,0)
44h	2	00 00	0 0	Padding for 4 byte alignment (Could be a value other than zero)

BMP Header Structure

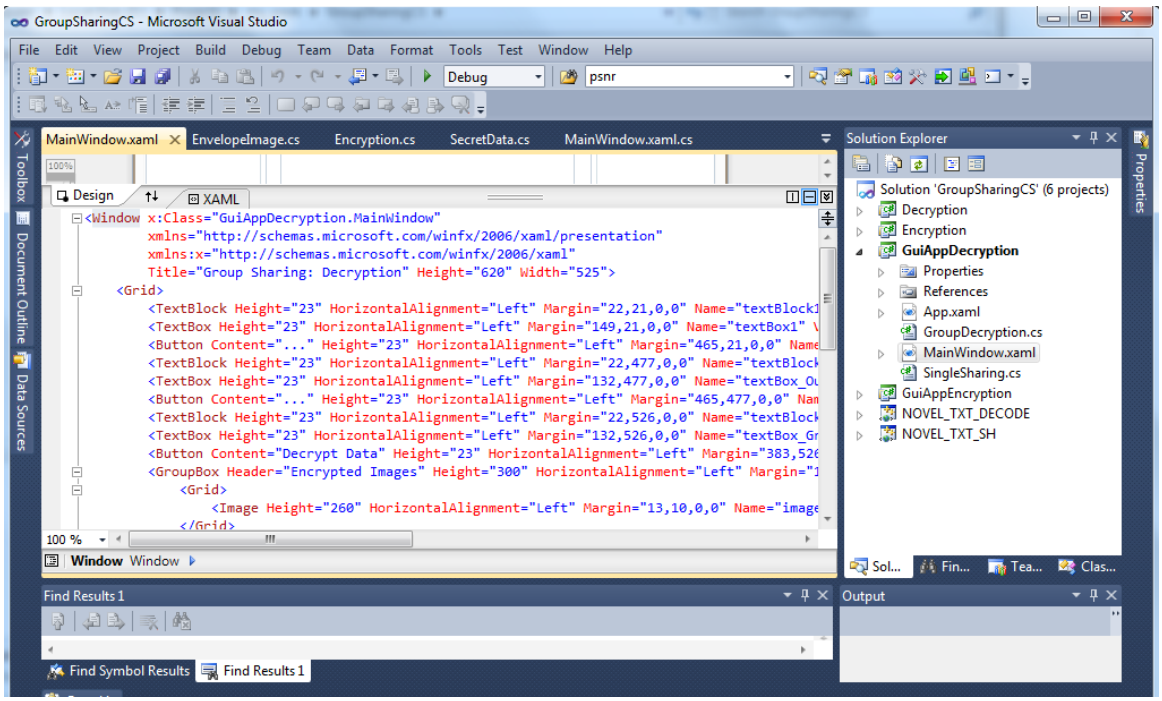
C.1.3 BMP Header Structure in Programming Concept

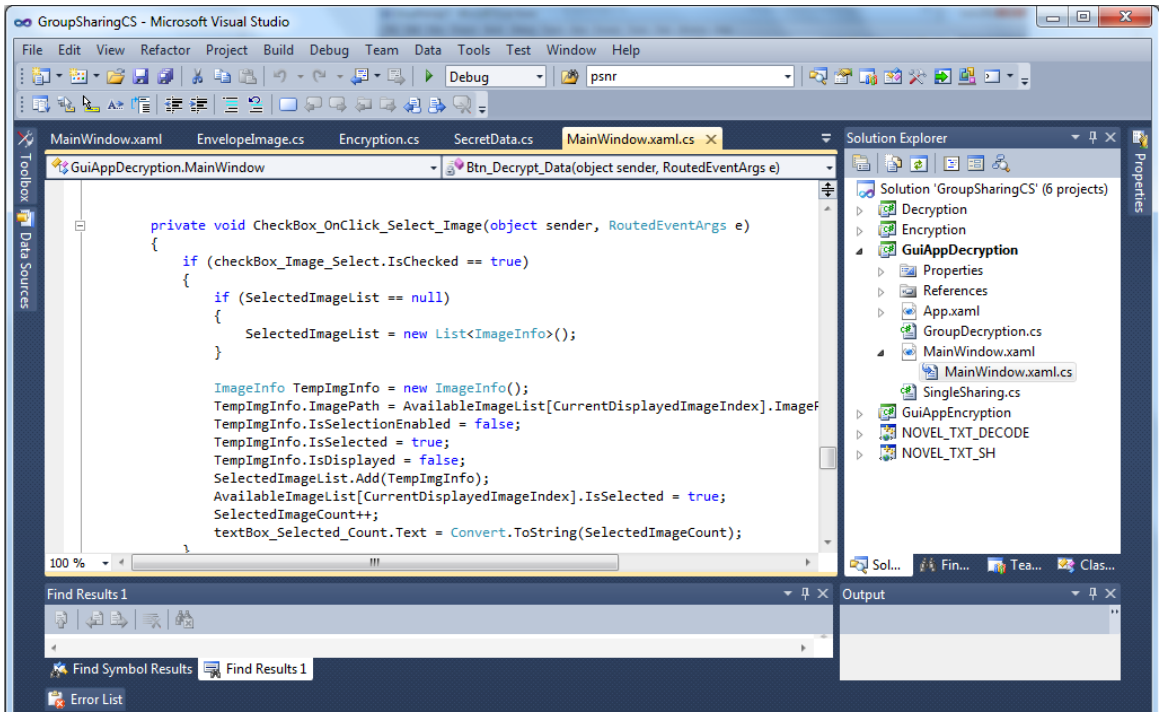
```
typedef unsigned char u8;
typedef unsigned short int u16;
typedef unsigned int u32;
typedef struct bmp_hdr {
        u16     signature;      // Magic Number (unsigned integer 66, 77)
        u32     file_size;      // Size of the BMP file
        u16     reserved1;      // Application Specific
        u16     reserved2;      // Application Specific
        u32     offset;         // The offset where the bitmap data (pixels) can be found.
        u32     info_hdr_size;  // The number of bytes in the header (from this point).
        u32     width;          // The width of the bitmap in pixels
        u32     height;         // The height of the bitmap in pixels
        u16     pannel;         // Number of color planes being used.
        u16     bpp;            // The number of bits/pixel.
        u32     compression;    // BI_RGB, No compression used
        u32     size_of_image_data;     // The size of the raw BMP data (after this header)
        u32     hor_reso_per_meter;  // The horizontal resolution of the image
        u32     ver_reso_per_meter;  // The vertical resolution of the image
        u32     colors_in_palette;      // Number of colors in the palette
        u32     imp_colors_used;        // Means all colors are important
}BMP_HEADER;
```

Appendix D

D.1 GUI Designing Phase

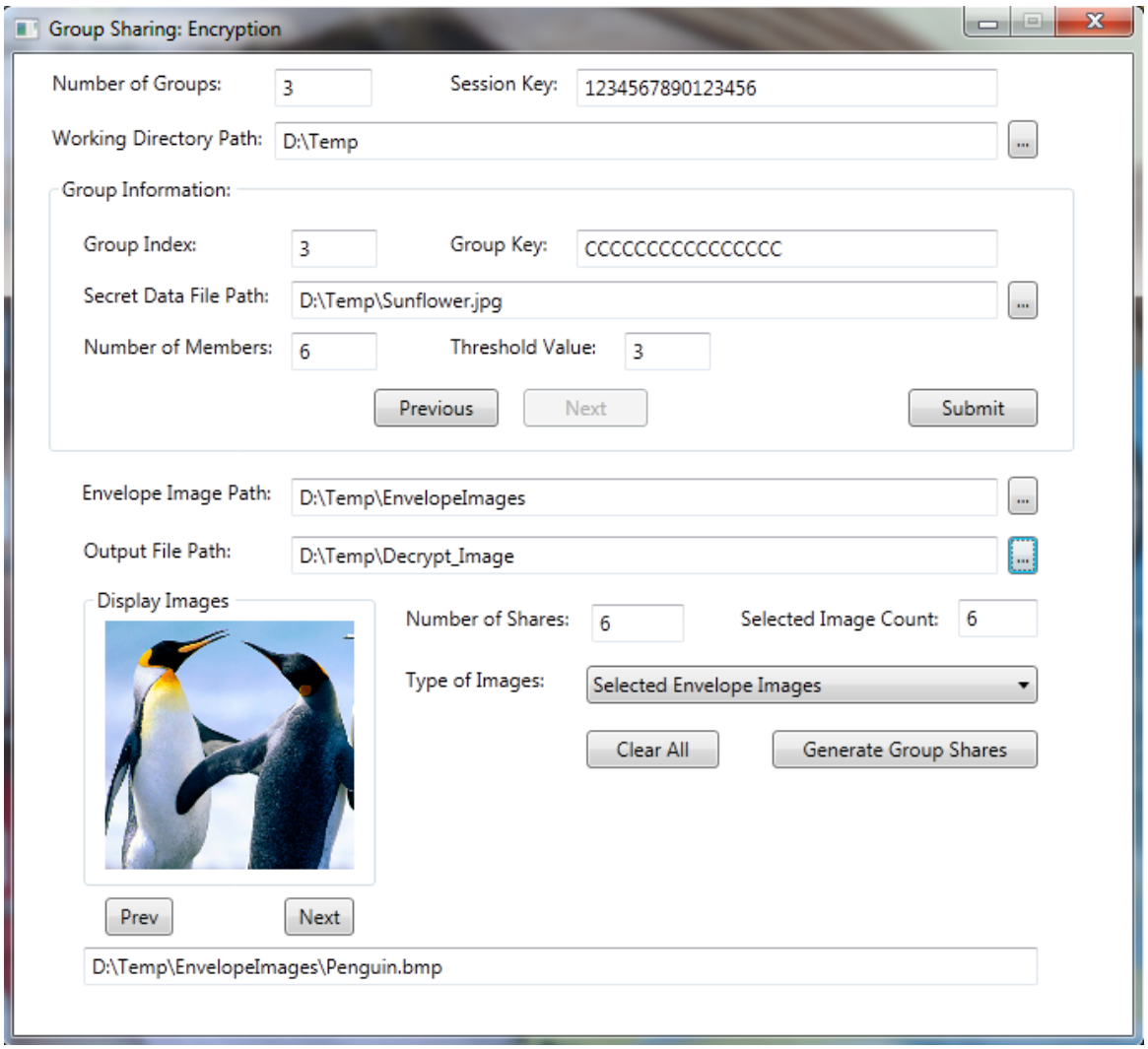
We have used Microsoft .NET WPF framework for designing. This is XAML based GUI design and code is written in C#. This GUI will work properly in .NET 4.0 and above version. Following snapshot shows Visual Studio environment for project development.





D.2 GUI Designed for Group Sharing Encryption

Following Snapshot shows the GUI for application that performs end to end encryption operation. Here we can set number of groups and also provide all the information required (Group key, Secret data for the group, Number of members in a group and Threshold value for that group) for each group to generate intermediate shares. Once intermediate shares are generated, this application provides options to select envelope images and insert the encrypted intermediate shares into it. Once steganography images are generated, we can view the final shares using this application GUI.



D.3 GUI designed for Group Sharing Decryption

Following snapshot shows GUI for the application that performs end to end decryption process. This application expects input as encrypted image path where all the required steganography images are available. Here user can select desired number of steganography images that can be used for reconstruction of secret. This GUI also provides an option to insert an appropriate group key and an output folder location where the decrypted secret will be stored.